*Федеральное агентство по образованию РФ*

*Государственное образовательное учреждение*

*высшего профессионального образования*

*«Московский государственный технический университет им Н.Э. Баумана»*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Факультет информатики и систем управления*

*Кафедра ИУ-1*

Лекции

По курсу «ВМСиС»

«Вычислительные машины системы и сети»

*Преподаватель Внуков Андрей Анатольевич*

awnkw@mail.ru

Оглавление

[Список используемых сокращений 2](#_Toc1156899)

[1. Вычислительные архитектуры на СБИС (ВА на СБИС). Классы алгоритмов. 3](#_Toc1156900)

[1.1. Универсальные процессоры 3](#_Toc1156901)

[1.2. Обзор современных ВА на СБИС 13](#_Toc1156902)

[1.3. Полное пространство алгоритмов 16](#_Toc1156903)

[1.4. Описание современных ВА на СБИС 20](#_Toc1156904)

[1.5. Элементная база ВА на СБИС 28](#_Toc1156905)

[2. Особенности обработки информации в вычислительных архитектурах 30](#_Toc1156906)

[2.1. Параллельная обработка информации в вычислительных архитектурах на СБИС 30](#_Toc1156907)

[Разработка АПК 31](#_Toc1156908)

[2.3. Критерии эффективности 38](#_Toc1156909)

[2.4. Ресурсы вычислительной системы 41](#_Toc1156910)

[Процессоры 41](#_Toc1156911)

[3. Способы реализации вычислительных архитектур 44](#_Toc1156912)

[3.1. RISC– и CISC–процессоры 44](#_Toc1156913)

[Типы команд 44](#_Toc1156914)

[Фазы обработки команд 46](#_Toc1156915)

[Обстоятельства возникновения RISC-процессоров 50](#_Toc1156916)

[CISC-процессоры 52](#_Toc1156917)

[Области применения RISC- и CISC- процессоров 52](#_Toc1156918)

[3.2. Компоненты информационных технологий 54](#_Toc1156919)

[Аппаратное обеспечение 54](#_Toc1156920)

[3.2.1 Фон-неймановская вычислительная архитектура 54](#_Toc1156921)

[3.2.2 Гарвардская вычислительная архитектура 56](#_Toc1156922)

[Классификация ВС по Флинну. 63](#_Toc1156923)

[5.ВА на СБИС. 68](#_Toc1156924)

[5.1. Разработка и использование архитектурных концепций ВС на СБИС. 71](#_Toc1156925)

[5.2. Краткая характеристика ВА на СБИС. 72](#_Toc1156926)

[5.3. Принцип систолической обработки данных 82](#_Toc1156927)

[5.3.1. Применение систолических полей 85](#_Toc1156928)

[5.4. Задача перемножения матриц 94](#_Toc1156929)

## Список используемых сокращений

CISC – **Complex** **instruction** **set** computing,или **complex** **instruction** **set computer**  (компьютер с полным набором команд)

FL – пакет Fuzzy Logic(инструментальное средство разработки нечетких объектов)

MIPS – миллион инструкций в секунду

MFLOPS – миллион операций с плавающей точкой в секунду

RISC –  restricted (**reduced**) **instruction** **set** **computer** (компьютер с сокращённым набором команд)

АЛУ – арифметико-логическое устройство

АПК – аппаратно-программный комплекс

АЦП – аналогово-цифровой преобразователь

БИС – большая интегральная схема

ВА – вычислительная архитектура

ВКИП – выборка команд из памяти

ВС – вычислительная система

ВУ – внешнее устройство

ГВА – гарвардская вычислительная архитектура

ДК – дешифратор команд

ИЛ – интерфейс линков

ИС – информационная система

К – коммутатор

ЛОЗУ – локальное оперативное запоминающее устройство

МГВА – модифицированная гарвардская вычислительная архитектура

МК – микроконтроллер

МП – микропроцессор

МПИ – мультипроцессорный интерфейс

ОЗУ – оперативное запоминающее устройство

ООЗУ – общее оперативное запоминающее устройство

ОС – операционная система

ОШ – общая шина

ПВЛ – процессор виртуальных линков

ПВВ – процессор ввода-вывода

ПДП – прямой доступ в память

ПЛИС – программируемая логическая интегральная схема

ПМ – программный модуль

ПО – программное обеспечение

ППТ – процессор плавающей точки

ПЭ – процессорный элемент; ПС – процессорная сеть

ПЯ – процессорное ядро

РСК – регистр состояния канала

СБИС – сверхбольшая интегральная схема

СГВА – супергарвардская вычислительная архитектура

СВТ – средства вычислительной техники

СС (EMI) – система связи с внешней памятью, внешний параллельный интерфейс ОЗУ

ССК – слово состояния канала

СУ – система управления

Т – транспьютер

ТП ЦСП – транспьютероподобная цифровая сигнальная система

УБС – устройство барабанного сдвига

ЦАП – цифро-аналоговый преобразователь

ЦМК – цифровой микроконтроллер

ЦП – центральный процессор

ЦСП – цифровой сигнальный процессор

ША – шина адреса

ШВВ – шина ввода/вывода

ШД – шина данных

ШУ – шина управления

# 1. Вычислительные архитектуры на СБИС (ВА на СБИС). Классы алгоритмов.

## 1.1. Универсальные процессоры

В настоящее время на рынке имеется целый ряд процессоров, которые определяются как универсальные процессоры. Их универсальность понимается, в том смысле, что они могут реализовывать практически любые задачи. Эти процессоры обладают развитой системой команд, для них создано системное программное обеспечение, позволяющее реализовать их действительно широкие возможности.

Эффективность использования ПЛИС заключается в том, что можно сократить время вычисления и задачу можно реализовать в виде параллельного алгоритма.

Процессоры различаются внутренней архитектурой, которая накладывает определенные ограничения на их эффективное применение.

Виды универсальных процессоров:

* Транспьютеры (Т)
* Цифровые сигнальные процессоры (ЦСП)
* Транспьютероподобные ЦСП (ТП ЦСП)
* Микроконтроллеры (МК)
* Универсальные Микропроцессоры (МП)

Области применения: транспьютеры – массово-параллельные вычислительные системы со значительным вычислительным ресурсом по: производительности, по объему оперативной памяти и пропускной способности каналов передачи данных.

ЦСП предназначены для прямой обработки электротехнических сигналов.

ТПЦСП имеют более высокие современные технические характеристики по сравнению с транспьютерами. Область применения та же.

Микроконтроллеры используются для программного управления техническими устройствами и объектами. Применяются для автоматизации промышленных установок.

Универсальные микропроцессоры применяютсяя в различных средствах вычислительной техники: ПК, ноутбуки, серверы.

Современные МП являяются многоядерными.

Все перечисленные процессорные элементы позволяют реализовывать вычислительные архитектуры с последовательным и параллельным выполнением команд и с последовательной и параллельной обработкой данных.

Для таких вычислительных архитектур в настоящее время эффективной являяются также элементная база ПЛИС. Они выгодно отличаются от СБИС, позволяя реализовывать не один конкретный аппаратный алгоритм обработки данных, а большое количество путем перепрограммирования.

Эти ПЭ используются в универсальных и специализированных ВА, а также могут использоваться в однородных и неоднородных процессорных сетях.

Однородная процессорная сеть – ПС, которая состоит только из ПЭ одного семейства (например, только из Т). В каждом ПЭ находятся одинаковые устройства внешних коммуникационных каналов – линков. Вместо Т можно поставить ТП ЦСП (у них тоже есть линки), которые представляют собой более современную элементную базу.

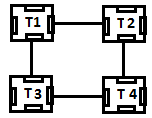


Рис. 1 Однородная процессорная сеть

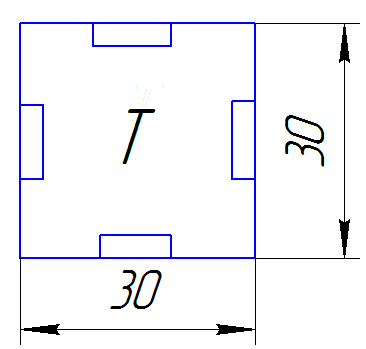


Рис. 2 Размеры чипа обычно не превышают 30 мм.

При этом площадь самого кристалла обычно не превышает квадратного сантиметра, остальную площадь занимают 84 штыревых контакта.

При соединении определенных интерфейсов получаются различные типовые структуры:

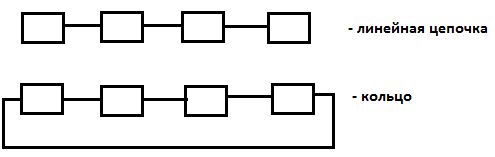


Рис. 3 Структуры линейная цепочка и кольцо

Неоднородная процессорная сеть – ПС, когда используются различные семейства ПЭ и они имеют различные интерфейсы и не могут с друг другом взаимодействовать. Например, для соединения семейства Т нам потребуется интерфейсная схема.

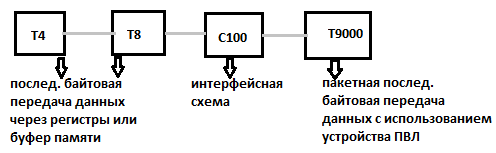


Рис. 4 Объеденинение транспьютеров разных семейств в одну сеть.

ПВЛ – процессор виртуальных линков.

Однородная ПС более дешева и проще устроена. Неоднородная ПС требует наличие дополнительных устройств, что более затратно.

* [**Транспьютеры**](http://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) (Т)

(transputer = transistor + computer) – универсальные микропроцессоры (МП) с RISC-процессором на кристалле, несколькими интерфейсами для внешней передачи данных, ОЗУ на кристалле, интерфейсом доступа к внешней памяти, процессором плавающей арифметики, системным устройством на кристалле, аппаратным планировщиком процессов, реализованном микропрограммой на кристалле, двумя таймерами.

|  |
| --- |
| **Рис. 5.** 32 битный транспьютерный чип IMST414B (1989) |

Первый транспьютер был разработан Inmos LTD(UK,Bristol).

Имеет специальную микропрограмму, которая позволяет выполнять работы в режим реального времени. Самое важная часть транспьютера – внешние коммуникационные каналы – «линки» («links»), которые на расстояние до 30 см могут передавать данные (сигнал).

Таймер процессов высокого приоритета добавляет единицу в свой регистр каждую микросекунду (1 мкс) и рассчитан на 4297 сек (приблизительно 1 час 12 минут) работы без переполнения 32х битного регистра таймера, а таймер процессов низкого приоритета добавляет единицу в свой регистр каждые 64 мкс. Это сделано для того чтобы, таймер высоко приоритетных процессов и таймер низко приоритетных процессов определяет уровень процессов. Процессы высокого приоритета могут прерывать процессы низкого приоритета. Если процесс высокого приоритета будет много или их очень много, то может не дойти до процессов низкого приоритета. Процессы низкого приоритета предназначены для приема, передачи и обработки данных. Процессор высокого приоритета выполняются только при наступлении определенного события, например, в случае возникновения аварийного состояния. Чтобы поддерживать высокие процессы – надо быстро обрабатывать регистр, а задачи в низком приоритете более сложные и выполняются долго, сделан длинный 32 битный таймер. После определенного времени (~76.8 часа) при переполнении разрядной сети, мы возвращаемся во времени назад (отбрасывается 33ий разряд). Поэтому два таймера нужны для согласованного одно и того же времени и выполнении процессов высокого и низкого приоритета.

Назначение транспьютеров как процессорных элементов (ПЭ) – создавать различные сетевые архитектуры, размещаемые на транспьютерных платах (сети процессорных элементов находящихся на плате). Они эффективны для реализации систем с массовой параллельной обработкой данных. А также вычислительные системы с массовым параллелизмом и однородные массовые системы. Для однородных систем не требуется дополнительных вычислительных устройств.

Несмотря на то, что транспьютер относится к универсальным процессорным элементам, с его помощью можно реализовать далеко не все классы алгоритмов с требуемой эффективностью. Можно сказать, что только некоторые классы алгоритмов имеет смысл реализовывать с помощью транспьютерной технологии.

Транспьютеры обладают внешними коммуникационными каналами, на основе которых можно создавать системы.

Основные ресурсы транспьютера:

1. Суммарная производительность
2. Суммарная память
3. Пропускная способность всех внешних каналов данных

Первый транспьютер был создан в 1985 году. INMOS Ltd (ныне - подразделение [STMicroelectronics](http://ru.wikipedia.org/wiki/STMicroelectronics)) изобрела и выпустила устройство (процессор) – транспьютер (Transistor + Computer = Transputer).

**Существуют 2 разновидности процессора**:

1. RISC (Reduced Instruction Set Computing) – процессор – это процессор с сокращенным набором команд. Используется в гарвардской архитектуре;
2. СISC (Complex Instruction Set Computing) – процессор – это процессор с полным набором команд. Используется в фон-неймановской архитектуре.

Т имеет 4 уникальных линковых интерфейса, которые обеспечивают корректную передачу данных на расстоянии до 30 см.

**Аппаратный планировщик** (АП) – специальная программа, микрокод которой записан на кристалле, позволяющий одновременно поддерживать выполнение практически неограниченного количества вычислительных процессоров. Их количество определяется только объёмом оперативной памяти. АП использует опыт таймера Т.

АП взаимодействует с двумя таймерами на кристалле, которые работают с процессами высокого (ПВП) и низкого (ПНП) приоритетов.

Вектор прерывания состоит из адреса программы обработки прерывания и слова состояния процессора.

В Т используются только ПВП и ПНП. 1 бит отводится в Т на указание приоритета процесса.

Основная масса процессов программного обеспечения – ПНП.

ПВП – процессы, которые должны быть достаточно короткими по времени выполнения, чтобы давать выполняться массе процессов с низким приоритетом (например, процессы, сигнализирующие об аварийной ситуации).

Таймер ПВП – 1мкс -> добавляет +1 в регистр состояния таймера ПВП

Таймер ПНП – 64 мкс -> добавляет +1 в регистр состояния таймера ПНП.

ПО Т поддерживает такую вычислительную модель, в которой ПВП если выполняется, то должна выполниться до конца, если только он может выполниться до конца.

На выполнение каждого ПНП дается 1 квант времени, равный 2\*1024 мкс. 500 раз может быть запущены на выполнение разные ПНП.

**В RISC архитектуре существуют 3 группы команд:**

1. Команды, которые обрабатывают данные, имеющиеся только в регистрах ЦП
2. Команды, которые берут данные из памяти и записывают в регистр ЦП
3. Команды, которые берут данные из регистра ЦП и записывают в память

**В СISC архитектуре существуют группы команд:**

1. Перечисленные простые группы комнад RISC архитектуры
2. Сложные команды, позволяющие одной командой обрабатывать большое количество данных и получать скалярный или векторный результат. Пример – скалярные и векторные умножения.

Если какой-то процесс не выполняется за указанный квант времени, то необходимо сохранить промежуточные результаты, запомнить состояние и информационные ошибки.

Согласно стандарту IEEE – 754 все операции делятся на:

1. Правильные (те, которые выполняются)
2. Неправильные (те, которые приводят к некорректным результатам)

Из этого стандарта вытекает требование, что вычисление не прекращается ни в каком случае и даже если операция неправильная она продолжается. Другими словами, вычислительный процесс будет идти и будет продолжаться – это важно для многих приложений, связанных с управлением в технических системах.

**Возможны 3 случая при выполнении процесса**:

1. Процесс полностью выполнился за время, меньшее одного кванта времени;
2. Процесс не выполнился за время, меньшее одного кванта времени и он мог бы ещё продолжаться;
3. Процесс низкого приоритета выполняется и данные не иссякли или не полностью обработаны.

В первом случае процессор просто берёт следующий процесс.

Во втором случае процесс должен через некоторое время возобновить свою работу и здесь используется механизм очереди процессов ПНП и ПВП. Если процесс новый, не начал выполняться и для него известны данные, то он становится в конец очереди.

В третьем случае процесс должен пристыковаться к концу очереди.

Активными очередями называются очереди активных процессов низкого и высокого приоритета, то есть имеются готовые данные (завершен ввод данных в процессор, вывод данных и наступил момент времени, с которого он может выполняться на процессоре).

Все остальные процессы называются пассивными.

Т может одновременно выполнять большое количество ПНП и ПВП. На каждом Т поддерживается псевдопараллельное выполнение команд. Каждый из процессов может делиться на подпроцессы.

Для выполнения последовательных и параллельных алгоритмов разработан язык Occam. Т возник в результате следующих этапов разработки:

1. Теории конкурентных процессов
2. Разработки языка конкурентного программирования Occam

**Конструктор SEQ** (последовательное выполнение процессов):

SEQ

<процесс 1>

. . .

<процесс N>

**Конструктор PAR** (параллельное выполнение процессов):

PAR

<процесс 1>

. . .

<процесс N>

Бывают случаи, что нужно всегда начать выполнение параллельного конструктора с первого процесса. В этих случаях используется оператор PRI PAR. Однако результат будет одинаковым.

PRI PAR

<процесс 1>

. . .

<процесс N>

Конструкторы определяют порядок выполнения процессов с использованием приоритета процессов и квантования времени на процессоре.

Важным конструктором является **конфигурационный конструктор**, который физически привязывает процессы к конкретному Т, а логические или программные каналы данных привязывает к физическим каналам передачи данных.

**Конфигурационный конструктор состоит из следующих конструкторов:**

1. Первый определяет тип Т
2. Второй определяет соответствие логических каналов физическим каналам
3. Третий определяет размещение процессов по Т

* [**ЦСП**](http://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D1%81%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80)

– цифровые сигнальные процессоры (DSP – Digitel Signal Prosessor) предназначены для цифровой обработки электротехнических сигналов, которые могут быть введены в эти процессоры, с помощью на кристальных устройств аналого-цифрового ввода, а выведены с помощью устройств цифро-аналогового вывода.

ЦСП делятся на несколько групп:

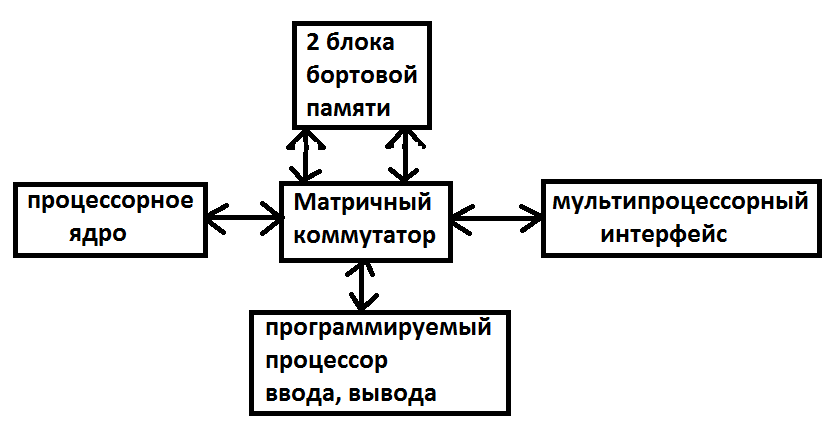
* Однопроцессорные ЦСП.
* Многопроцессорные ЦСП (2,3,5) (объединяющиеся в сети с помощью внешних линков связи, похожих по принципу работы на транспьютеры)
* Транспьютероподобные ЦСП (ТП ЦСП)
* Вычислительные ЦСП (используются только для вычислений)

Для всех ЦСП характерно, что производительность операций с плавающей точкой выше, чем производительность с целочисленными операциями. Это требование было положено в техническое задание при разработке ЦСП.

Данное требование возникает из режима реального времени, в котором должны обрабатываться практически любые электротехнические сигналы. Для реализации этого задания на кристалле ЦСП размещено устройство – аппаратный умножитель.

Целочисленные данные используются в задачах обработки изображений. В остальных задачах – данные с плавающей точкой.

Современные ЦСП 4-го поколения построены на принципах Супер-Гарвардской архитектуры (рис.6, \*на рисунке 2 блока двухпортовой памяти(изменить рисунок!)), в которой матричный коммутатор объединяет в единую вычислительную систему процессорное ядро, два блока двухпортовой памяти, мультипроцессорный интерфейс и процессов ввода-вывода со своим адресным пространством.



*Рис. 6 Супер-Гарвардскаяя ВА*

- подключение внешнего ОЗУ

- обмен данными между ЦСП

- параллельный байтовый интерфейс

*Программируемый процессор ввода, вывода:*

- можно осуществить ввод, вывод данных через внешние каналы линки связи

- можно подключить внешние устройства

- последовательный байтовый интерфейс

- Программируемые каналы прямого доступа в память

*Процессорное ядро:*

- ЦП

- аппаратный умножитель

- устройство барабанного быстрого сдвига влево-вправо

- буфер команд

- 2 генератора адресов данных в каждое ОЗУ

- таймер

*Матричный коммутатор:*

- передача данных, команд и управляющих сигналов между устройствами блоков

- уменьшение площади кристалла на коммуникации за счет уменьшения количества шин, соединяющих отдельные устройства напрямую и перенаправляющих данных между устройствами через коммутатор (подобно АТС)

В составе общей шины ЦСП имеются шины адреса и данных, работающие только с каждым ОЗУ. Это является одним из принципов внутренней архитектуры ЦСП (каждая память имеет индивидуальные каналы передачи адресов/данных)

Если ЦСП работают для обработки сигналов – сначала преобразовывается сигнал в код, в число с плавающей точкой, далее выдается численное значение, которое преобразуется в сигнал(ы).

Однопроцессорные ЦСП используются в мобильных средствах связи.

Многопроцессорные ЦСП имеют на кристалле 2,3,5 процессоров, которые выполняют операции с плавающей и фиксированной точкой. Процессоры с фиксированной точкой эффективны для обработки изображений (работают с целочисленными данными), процессоры с плавающей точкой необходимы для обработки звуковых и радиочастотных сигналов (звук, радиолокация). Некоторые ЦСП имеют на кристалле видеоконтроллер, кодек (звуковой ввод/вывод). Эти ЦСП могут реализовывать SIMD- и MIMD-алгоритмы, к которым относятся синхронная и асинхронная обработка потоков команд и данных.

SIMD – одна команда выполняется над большим количеством потоков данных

MIMD – много команд связаны с большим количеством потоков данных. (Более подробнее читайте «Классификация ВС по Флинну»)

\* Есть RISC- и CISC-процессоры, отличающиеся набором команд, последний более широк. Исторически при развитии процессоров, вводились новые более сложные команды (например: скалярное умножение двух векторов, умножение матриц). ЦСП построены, как RISC-процессор, поэтому все команды из системы команд выполняются за один такт. Например, команда умножения с накоплением c=a\*b+c, выполняется за один такт.

Вычислительные ЦСП имеют на кристалле только вычислительный ресурс и осуществляют исключительно цифровую обработку сигналов.

Операции с плавающей точкой выполняются быстрее, чем с фиксированной точкой.

Связано с тем, что есть аппаратный цифровой умножитель. Например, на ЦСП фирмы Analog Devices скорость работы с разным типом арифметики (с фиксированной и плавающей точкой) отличается в два раза.

Основные производители универсальных ЦСП со стандартным набором устройств и стандартными характеристиками, широко используемых в технических системах:

* [Analog Devices](http://www.analog.com/ru/index.html)
* [Texas Instruments](http://ru.wikipedia.org/wiki/Texas_Instruments)
* [Motorola](http://ru.wikipedia.org/wiki/Motorola)

Фирмы-производители бытовой техники производят свои собственные ЦСП с характеристиками достаточными для встраивания их в бытовую технику и устройства, например Panasonic (Matsushita Electric Industrial Co.)

Основные недостатки ЦСП:

* Пропускная способность (не все ЦСП имеют одинаково высокую пропускную способность внешних коммуникационных каналов). Например, самая высокая у [Texas Instruments](http://ru.wikipedia.org/wiki/Texas_Instruments)
* ***ТП ЦСП***

– транспьютероподобные ЦСП. ТПЦСП являются современным аналогом транспьютеров. Они построены на принципах ЦСП и используют преимущества транспьютеров и ЦСП. Они разработаны фирмами-производителями ЦСП и заполняют нишу транспьютеров с современными техническими характеристиками. В отличие от транспьютеров, которые имеют только локальные ОЗУ и обмениваются данными только через линки, ТПЦСП могут обмениваться данными через внешние каналы передачи данных, линки связи и через общую память. Скорость обмена через память выше, чем через линки, но через память сложнее организовать корректную передачу данных. С этой целью используются различные методы, например, метод «семафора» – один процессор допускается к памяти, остальные ждут своей очереди.

Распараллеливание по функциональному признаку – при добавлении новой функциональной возможности в техническую или информационную систему устанавливается новый процессорный элемент с программным обеспечением, поддерживающим выполнение этой функции синхронно или асинхронно с раннее введенными в систему процессорными элементами, каждый из которых обеспечивает независимое выполнение каждой функции. Такие системы строятся на принципах синхронной и асинхронной обработки каналов данных.

С помощью ТП ЦСП строятся системы с массовой параллельной обработкой данных. Их основными преимуществами являются компактность габаритов, малый вес, меньшее энергопотребление при высокой производительности и пропускной способности каналов передачи данных, высокая надежность.

(дописать: чем занимается каждый процессор; аппаратный умножитель; радио- и эхолокация; синхр. обработка команд и данных)

* **Микроконтроллеры** (МК) обладают достаточным ресурсом для реализации систем управления. Они, как и ЦСП, могут иметь на кристалле АЦП и ЦАП, но их вычислительные возможности значительно ниже, чем у ЦСП. Производительность операций с плавающей точкой ниже, чем у целочисленных операций (отсутствует аппаратный умножитель устройства сдвига данных).

Универсальные контроллеры представляют собой не отдельный чип, а плату, на которой имеется например, процессор фирмы Intel и различные устройства, обеспечивающие взаимодействие с объектами.

Стоимость микроконтроллеров, как правило, дешевле других универсальных процессоров.

По сравнению с универсальными МП, они имеют интерфейс на кристалле для ввода-вывода внешних аналоговых и дискретных сигналов, но обладают заметно меньшим ресурсом по производительности с монокристальными процессорами.

Наряду с МК существуют контроллеры, которые используют процессоры с Intel – совместимыми рядами, размещаемые на отдельной плате небольшого размера. На других платах находятся интерфейсы для работы с аналоговыми и дискретными каналами (рис.7).

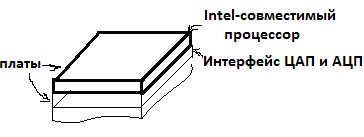


Рис. 7 Контроллер-Fastwell с Intel-процессором

Платы Контроллера-Fastwell, производства американской компании Origon, крепятся вертикально с помощью специального кронштейна, обладающие значительной механической жесткостью.

Как правило, МК в системах управления обеспечивают частоту ввода/вывода данных с датчиков на приводы от 10 до 100 Гц. Производительность их ЦП поддерживает такие возможности.

Преимущество такого контроллера – полная совместимость с ПО, разработанного на ПК. Однако при разработке ПО для контроллера следует учитывать отсутствие реальных аналоговых и дискретных каналов. При разработке можно использовать заглушки для работы с такими каналами или программные имитаторы, имитирующие работу с соответствующим типом канала.

* Универсальные МП – это процессорные элементы фирмы Intel, AMD и др. производителей, выпускающих совместимые процессоры, которые предназначены для реализации универсальных вычислительных архитектур (ВА). Такие архитектуры позволяют программно реализовывать наиболее широкий класс алгоритмов. Для них созданы операционные системы, системы программирования, реализована СУБД, Возможна реализация большого числа алгоритмов, в том числе может быть осуществлена эффективная эмуляция, позволяющая имитировать и моделировать работу выч. Систем, построенных на вышеперечисленных типах ПЭ.

Эффективность реализации алгоритмов в целом высокая, однако, уступает эффективности реализации специфических алгоритмов, которые м.б. реализованы лучше на других перечисленных универсальных процессоров.

Т.е. лучше узкоспециализированный алгоритм реализовывать на своем типе процессорных элементов. На универсальных процессорах можно реализовывать все вышеперечисленные алгоритмы.

Время параллельного умножения матриц размерности (n x n):

T=1\*t + round(n/2)

Этот параллельный алгоритм производит на первом шаге все независимые вычисления, на следующих тактах выполняются операции суммирования, получение промежуточных результатов сложения.

Ускорение u=tпослед/tпарал= n3\*(t\*+t+)/( t\*+ round(n/2)\*t+)

t\* -время выполнения операции умножения

t+-время выполнения операции сложения

В перечисленные процессоры можно добавить медийные процессоры, ориентированные на мультимедиа приложения, систолические процессоры, предназначенные для реализации систолической ВА на СБИС, для аппаратной реализации алгоритмов. Может быть достигнуто ускорение в 103 раз. Среди систолических процессоров имеются простые и сложные программируемые процессоры, которые отличаются от простых наличием памяти.

Простые имеют только регистры и выполняют только определенные операции с данными в этих регистрах, например, операцию умножения с накоплением, операцию нахождения среднего по нескольким операндам. Сложные программируемые процессоры в их память записывается как правило программный цикл, который повторяется один раз.

Особенностью медийных процессоров является то, что они способны быстро выполнять наиболее часто встречаемые операции по обработке изображений и звука с высокой производительностью. Для этого в систему команд введены специальные команды.

Для аппаратной реализации нейроалгоритмов используется специальный класс процессоров -нейрочипы, которые также не относятся к универсальными МП, но позволяют наиболее эффективно реализовать нейроалгоритмы с максимальным быстродействием. Однако стоимость нейрочипов самая высокая среди элементной базы, предназначенной для параллельной обработки информации, а системные инструментальные и прикладное ПО является весьма дорогим и требует специальной подготовки разработчиков, которое также является платным.

Дополнительным аспектом является необходимость трансформация исходного последовательного алгоритма в параллельный нейроалгоритм, который реализуется в аппаратном обеспечении с использованием нейрочипов.

Другой важной особенностью использования нейрочипов является то, что эффективность их использования потребовало введения двух новых характеристик:

1. Количество соединений в единицу времени (количество передач данных в единицу времени) между отдельными компонентами нейросети, т.е. между слоями.
2. Показатель длительности обучения нейросети с конкретным нейроалгоритмом.

## 1.2. Обзор современных ВА на СБИС

В настоящее время существует элементная база для построения нескольких современных ВА на СБИС, позволяющих эффективно реализовывать определенный класс алгоритмов.

|  |
| --- |
| **Рис.8.** Современные ВА на СБИС |

Если точка принадлежит некоторой области, то это означает, что существует элементная база с двумя характеристиками – структурным размером и степенью интеграции.

Структурный размер представляет собой технологические нормы производства СБИС. Структурный размер фактически определяется толщиной соединений, площадью реализации транзистора или функционального блока. Минимальный структурный размер является производной от технологических норм производства, т.к. для увеличения количества устройств на кристалле стремятся минимизировать площадь, занимаемую активным элементом или функциональным блоком.

Степень интеграции определяется количеством активных элементов-транзисторов на кристалле СБИС. Активные элементы используются в ЦП, памяти (наибольшее количество), в общей шине, интерфейсах.

I – ВА память

II – ВА «логика + память» (логика + память+процессор)

Специализированные ВА на СБИС

III – систолическая ВА

IV – мультипроцессорная ВА

VI – нейровычислительная ВА

V – универсальная ВА на СБИС

Специализированные ВА на СБИС – характеризуется алгоритмическим обеспечением, которое узкоспециализировано для выполнения на этих 5 ВА на СБИС, причем алгоритмы, которые эффективно выполняются в одной ВА на СБИС, не будут столь же эффективно выполняться в других ВА на СБИС, возможно, их даже не удастся реализовать.

В универсальных ВА на СБИС могут быть выполнены любые алгоритмы, составляющие полное пространство алгоритмов, однако эффективность их выполнения не будет столь высокой, как в специализированных ВА. Возможно, потребуется больше дополнительный вычислительный ресурс в виде производительности, памяти, пропускной способности каналов передачи данных и потребуется больше времени для их выполнения.

Эффективность выполнения алгоритмического и программного обеспечения определяется, в том числе, и системами, средствами (например, компилляторами, которые позволяют оптимизировать загружаемый код с точки зрения занимаемой памяти и скорости выполнения). При этом неоптимизированный код программы на С будет уступать по времени выполнения той же программе, написанной на Асемблере, если исходный текст на Асемблере написан хорошо. Возможны два варианта написания программы на асемблере:

1. Программа с самого начала пишется на Асемблере
2. Делается дисасемблирование программного текста, написанного на С с опцией ручного дисасемблированияя. В этом случае удается создать более эффективный исходный текст программы.

Алгоритмическое обеспечение реализованное на специализированных ВА обладает большей эффективностью по сравнению с тем же алгоритмическим обеспечением, реализованным на неспециализированных ВА.

Назовем известные универсальные ВА на СБИС

* ПЭВМ на транспьютерах
* Универсальные ВА на СБИС, архитектура процессоров VLIW - very long instruction word (например, процессор Itanium 2001г.), связки команд выполняемых на 3 внутренних конвейерах конманд.
* Супер ЭВМ
* Квантовые компьютеры
* Компьютеры обратимых вычислений
* **Обратимые вычисления** - модель вычислений, в которой процесс вычисления является в некоторой степени обратимым. Например, в вычислительной модели, использующей наборы состояний и переходов между ними, необходимым условием обратимости вычислений является возможность построения [однозначного (инъективного)](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%8A%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) отображения каждого состояния в следующее за ним. На XX век и начало XXI века обратимые вычисления обычно относят к нетрадиционным моделям вычислений.
* Графические вычислители (TESLA, IBM и др.)
* ЭЛЬБРУС СУПЕР ЭВМ, ЭЛЬБРУС 90
* Целью Концепции является обоснование необходимости и возможности создания эксафлопных средств математического моделирования, обеспечивающих возможности технологического прорыва в следующих областях, определенных Комиссией при Президенте Российской Федерации по модернизации и технологическому развитию экономики России в качестве стратегических.

Таблица – Эксафлопные проекты

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | **США** | | **Евросоюз** | **Китай** | |
|  | | DOE – более 100 млн.$/год; NSF и университеты до 30 млн.$/год; DARPA UHPC около 75 млн.$ на 4 года; Всего на исследования без стоимости машин около 150 млн.$/год | | Еврокомиссия, программы PRACE, DEISA и EESI, национальные программы, университеты и индустрия.  Всего на исследования без стоимости машин –  29-43 млн. $ в год | 2012 г. – суперЭВМ петафлопного класса на собственной элементной базе  Открытая часть финансирования – 67 млн.$ | |
| **10 Пфлопс** | 2011-2012 г.г.  20 Пфлопс: Sequoia, Titan  10 Пфлопс: Mira | | 2012-2013 г.г.  SuperMUC –3 Пфлопс  Hermit –4-5 Пфлопс | | | 2015 г. -суперЭВМ 10 Пфлопс;  Бюджет: 107 млн. $ |
| **100Пфлопс** | 2015-2016 г.г. –несколько суперЭВМ (от 50 до 150 Пфлопс) | | Официальных  данных нет | | | 2015-2020 г.г.  Несколько суперЭВМ петафлопной производительности, хотя бы одна 50-100Пфлопс  Бюджет: 615 млн.$ |
| **1000 Пфлопс** | 2018-2019 г.г. –  несколько суперЭВМ  (от 1 до 2 Эксафлопс) | | 2020 г. –  европейскийЭксафлопс | | | 2020г.  1~10 Эксафлопс  Бюджет: еще неизвестен |
| International Exascale Software Project (IESP)  Международный проект по созданию программного обеспечения  для вычислительных систем эксафлопного класса  (США, Евросоюз, Япония, Китай, Россия и другие страны) | | | | | | |

* ПС2000

Разработана и реализуется российская программа по созданию суперкомпьютеров с производительностью 1018 к 2018 году.

## 1.3. Полное пространство алгоритмов

Полное пространство алгоритмов - это трехмерное пространство координатами, осями которого являются типы, на каждой из координатных осей имеется свой масштаб; каждый определенный объем может соответствовать одному из классов алгоритмов. Для эффективной реализации алгоритмов из этого класса может быть использовано определенная ВА на СБИС, которая, в свою очередь, эффективно реализуется на конкретных процессорных элементах и элементной базе. Рассматривать полное трехмерное пространство алгоритмов затруднительно, поэтому рассматривают частный случай. В частности проекции пространства алгоритмов на плоскости, например, тип размера – тип управления.

Типы пространства алгоритмов:

* Тип управления
* Тип размера
* Тип коммуникационной структуры

|  |
| --- |
|  |

Рис.9 Полное пространство алгоритмов

Типы управления:

* централизованный
* распределенный
* потоковый.

Типы размера вычислительного модуля:

* малый постоянный
* малый (средний) переменный
* большой переменный.

Типы коммуникационной структуры:

* Одномерная
* двухмерная
* трехмерная
* многомерная

*Рассмотрим тип управления:*

Централизованное – обязательно наличие главного модуля, через который осуществляется управление порядком работы вычислительных модулей.

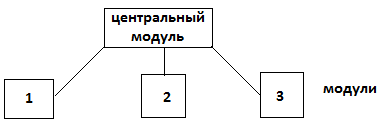


Рис.10 Централизованное управление

Распределенное – отсутствие главного модуля, распределение всех вычислительных модулей по вычислительному ресурсу.

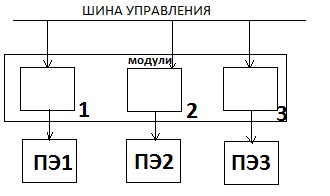


Рис. 11 Распределенное управление

Потоковое – управление вычислениями осуществляется данными или потоками данных.

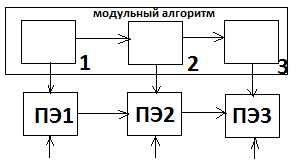


Рис. 12 Потоковое управление

Рассмотрим типы размера вычислительного модуля:

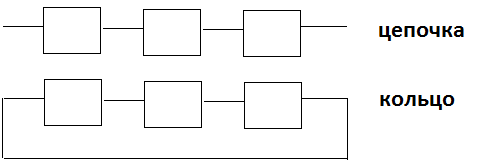
* - малый постоянный размер характерен для процессоров, не имеющих памяти или обладающих малым объёмом памяти
* - средний переменный размер характерен для процессоров с памятью незначительного объёма
* - большой переменный размер характерен для процессоров, которые могут быть укомплектованы значительным объёмом памяти

Единица измерения размера вычислительного модуля – количество операций на выч. модуль.

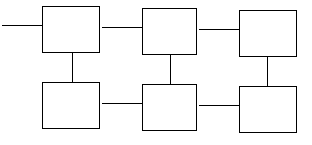
Единица измерения аппаратной поддержки для реализации выч. модуля – байт.

*Рассмотрим типы коммуникационной структуры*:

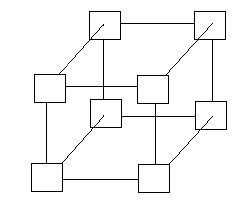
- Одномерная коммуникационная структура (цепочка и кольцо)



- Двумерная коммуникационная структура

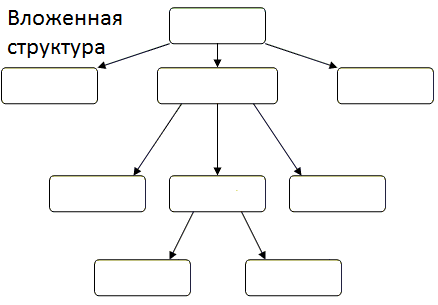
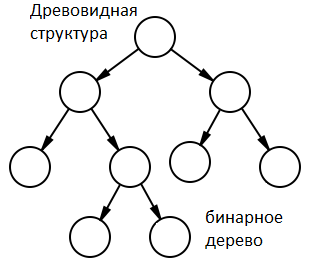


- Трехмерная коммуникационная структура



Широко используются древовидные и вложенные коммуникационные структуры:

Древовидная структура- это графы



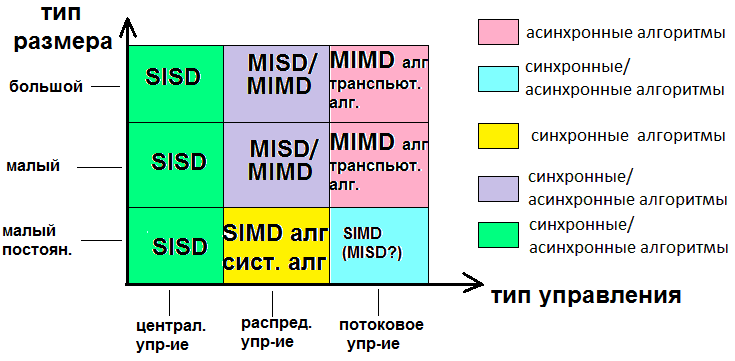
Бинарные(один вход, два выхода) и с полными связями(добавляются связи одного уровня)

фотка!!!!!!вложенная структура

Для реализации алгоритмов используются ВА на СБИС, которые в свою очередь реализуются с помощью универсальных и специализированных процессоров, а также элементной базы для создания различных однородных и неоднородных вычислительных коммуникационных структур (адаптеры, электронные коммутаторы, согласующие СБИС).

При аппаратной реализации применяются специализированные элементные базы, такие как ПЛИС, систолические процессоры, нейрочипы, а также ЦСП, ТП ЦСП и транспьютеры.

Выполним проекцию полного пространства алгоритмов на плоскость: тип размера – тип управления. При этом не вводятся единицы измерения. Подразумеваем их специфическими ранее введенным типам.



SIMD алгоритмы- систалические алгоритмы

Рис. 13 Проекция полного пространства алгоритмов на плоскость «упр-ие/размеры»

Сокращения

Классификация по Флину ВС и алгоритмов:

1. Класс SISD – фон-неймановская ВА
2. Класс SIМD – Гарвардская ВА
3. Класс МISD – конвейерные ВА процессоров и процессорных сетей
4. Класс МIМD – внутренняя ВА некоторых ЦСП, мультимикропроцессорная ВА, нейроВА

Классификация по Флину опирается на 2 признака ВС:

- количество потоков команд

- количество потоков данных

* Single Instruction Stream (SI) – один поток команд
* Multiple Instruction Stream (MI) – множественный поток команд
* Single Data Stream (SD) – один поток данных
* Multiple Data Stream (MD) – множественный поток команд

SISD – алгоритмы могут быть синхронными и асинхронными;

SIМD – всегда синхронные;

МISD – в сбалансированной ВС – синхронные, в несбалансированной – асинхронные;

МIМD – всегда асинхронные.

## 1.4. Описание современных ВА на СБИС

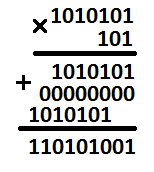
Современные ВА на СБИС представлены на рисунке 8 (стр.16). Этот рисунок имеет качественное значение. Отдельные области имеют взаимопересечения. Это говорит о том, что имеется элементная база, с помощью которой может быть реализована одна или несколько ВА на СБИС. Например, пересечение второй и третьей области означает, что ПЛИС могут быть использованы для реализации систолической ВА. Пересечение третьей и четвертой областей показывает, что транспьютеры могут быть использованы для реализации третьей и систолической ВА. Пересечение четвертой и шестой показывают, что транспьютеры могут быть использованы для реализации шестой нейровычислительной архитектуры. Пересечение шестой и пятой отражает возможность применения универсальных Intel-совместимых процессоров для реализации шестой нейровычислительной архитектуры. А пересечение первой и второй областей, в общем, является некорректным. Элементная база первой ВА используется при реализации пятой универсальной ВА.

Дадим краткую характеристику с точки зрения используемой элементной базы для ранее названных ВА на СБИС.

Для реализации ВА память (I) используются СБИС ОЗУ, предназначенные для реализации операций доступа к памяти и эффективных моделей алгоритмов работы с памятью (стековая, магазинная, общая, локальная, разделяемая и другие).

Умножение возможно одним из способов:

1. С помощью аппаратного умножителя
2. С помощью операций сдвига и сложения:



Элементная база: СБИС ОЗУ различных типов динамической и статической памяти, Эффективные модели памяти: стековая, магазинная, общая, локальная, разделяемая и другие. Модели памяти предполагают реализацию алгоритмов доступа к памяти. Например, для общей памяти необходимо реализация механизма корректного доступа к данным, что как правило предполагает придание разных приоритетов разным операциям доступа к памяти. Три операции – чтение, запись, перезапись: запись – наивысший приоритет, перезапись – такой же приоритет или несколько пониже (промежуточный), чтение – наименьший приоритет. Операция чтения предполагает копирование данного из ячейки данных в регистр ЦП. Операция записи переписывает содержание регистра ЦП в память. Операция перезаписи предполагает чтение данного из памяти с последующим обновлением содержимого памяти. Реализуем алгоритм, связанные с сохранием и получением из памяти.

Для ВА «логика + память» (II) используются программируемые логические интегральные схемы (ПЛИС), позволяющие создавать на основе стандартных структурных элементов конкретные вычислительные архитектуры, реализующие конкретные алгоритмы. В ПЛИС имеются память, логические структуры, структуры ввода-вывода. Двоичный код, записанный в память, создает цепочки структурных элементов, которые имеются в данном типе ПЛИС. Среди них логические структуры, вычислительные структуры с использованием регистров, сумматоров и структуры ввода/вывода. В каждой ПЛИС содержится определенное количество таких типовых структур, а двоичный код, записанный в память, определяет конфигурацию аппаратного вычислителя, реализуемого с помощью данной ПЛИС. В блоке памяти могут содержаться данные и в которых хранится информация о конфигурации внутренних элементов. По периметру кристалла ПЛИС расположены устройства ввода-вывода, которые позволяют вводить и выводить данные в разных форматах цифровых данных. Кроме этих устройств ввода-вывода на кристалле имеются устройства ввода, вывода дискретных сигналов и аналоговых и цифровых сигналов. На кристалле имеются также элементы синхронизации, конфигурируемо-логические блоки (КЛБ). Преимущество использования ПЛИС – возможность реализации различных алгоритмов путём создания на ПЛИС фактически аппаратной реализации последовательного или параллельного алгоритма обработки команд и данных.

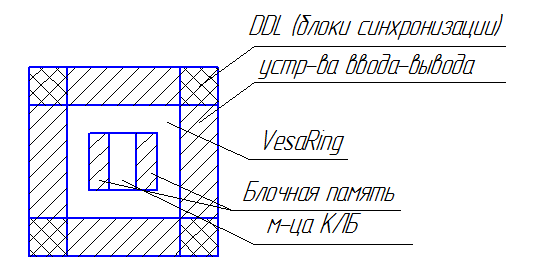
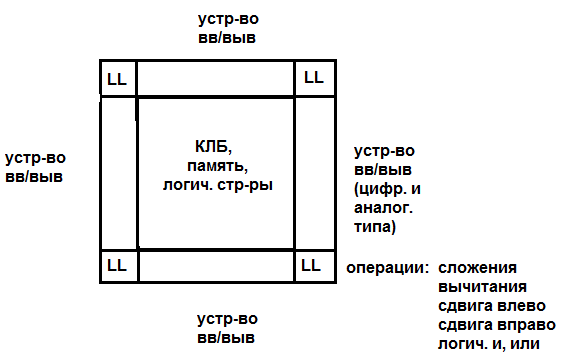


Рис. 14 Схема внутренней архитектуры ПЛИС FPGA

ВА логика+память предполагает создание аппаратного сопроцессора выполнения алгоритма. ВА логика+память+процессор позволяет реализовать на ПЛИС целиком и полностью хост-процессор (главный процессор) и аппаратную реализацию алгоритмического обеспечения в виде сопроцессора. Эта архитектура становится возможным при использовании ПЛИС 4-го поколения.

Для создания ПО для ПЛИС используются инструментальные системы программирования, позволяющие перевести вычислительный алгоритм в термины используемых операций ПЛИС. Это основные операции (сложение, вычитание, сдвиг влево, вправо, логическое и, или), а также операции ввода, вывода цифровых, аналоговых. Получаемый код загружается в ПЛИС и позволяет использовать только определенный набор вычислительных блоков. Эти блоки могут выполняться последовательно или параллельно одновременно. Таким образом, можно создать из исходного текста вычислительного алгоритма некоторые аппаратные цепочки на ПЛИС.

Теорема: Если существует алгоритм без бесконечных циклов, то его можно реализовать с помощью трех блоков, каждый из которых имеет один вход и один выход:

1. Последовательных блоков
2. Условного блока
3. Блока цикла

В ПЛИС существуют возможности для реализации алгоритмов, удовлетворяющих условию этой теоремы: 1) можно реализовать последовательность вычислительных блоков; 2) можно реализовать блок условий; 3) можно реализовать блок цикла.

Класс эффективных алгоритмов представляет собой аппаратные реализации операций с плавающей точкой, аппаратные реализации сложных операций, предполагающих выполнение нескольких команд, аппаратные реализации функций и подпрограмм, состояящих из операций и функция. Наиболее эффективная реализация достигается средствами только одной ПЛИС. Использование большего количества ПЛИС возможно,но передача данных между отдельными ПЛИС занимает больше времени чем надо.

В ВА логика+память реализованы новые команды, не входящие в общее количество команд ОС, возвращающие функции и подпрограммы, наборы данных.

В систолической ВА (III) используются систолические процессоры, ЦСП для реализации синхронно работающей процессорной структуры, в которой можно выделить две основные фазы: обработка данных и межпроцессорная передача данных. Каждая фаза обладает определенной продолжительностью, во время которой выполняется только одно действие - либо обработка данных, либо коммуникация. Данная вычислительная архитектура предназначена для «жесткой» аппаратной процессорной реализации конкретных алгоритмов.

Систолическая ВА позволяют реализовывать все операции, которые характерны для двух предыдущих ВА, и выполнять эти операции очень быстро.

Эта ВА способна выполнять операции, которые имеют реализацию в виде команд систолических процессоров (СП). Все СП делятся: 1) имеющие только регистры; 2) имеющие регистры и память. Первые реализуют как правило ограниченный набор команд (+,-,\*,/, пересылка данных). Вторые называются программируемые СП с большим набором команд как у любого универсального процессора, а в памяти могут храниться как программы, так и данные.

Существуют 3 фазы систолической обработки данных:

1. Фаза управления (2-3 мкс);

2. Фаза коммуникации (длительность 1 мс);

3. Фаза вычисления (длительность более 20 мс).

Различие в длительности фаз: фаза коммуникации/фаза управления ~500-1000 раз, фаза коммуникации/фаза вычисления ~20 раз (рис.15).

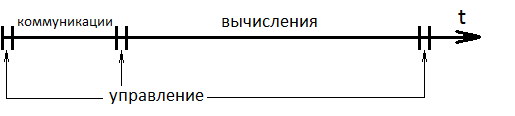


Рис15. Длительность фаз управления, коммуникации, вычисления.

Система управления систолическим вычислением строится с использованием программируемого таймера. Систолическая ВА использует 2 принципа параллельной обработки данных: 1. Конвейеризация вычислений с использованием одного или нескольких конвейеров 2. Высокая степень параллелизма обработки данных, которая означает физическое одновременное выполнение большого количества обработок данных во время фазы вычисления и большое количество передач данных во время фазы коммуникации.

Также как и «Логика+память» систолическая ВА выполняет и вычисление и операции с памятью. В отличие от «Логика+память» систолическая ВА осуществляет только синхронную обработку данных. Когда начало и окончание всех операций-коммуникаций и вычислений во всем процессорном поле синхронизировано.

Мультипроцессорная ВА (IV) реализуется на транспьютерах или ТП ЦСП. С помощью них строятся асинхронные вычислительные системы, состоящие из процессорных сетей, которые могут работать, в том числе и в режиме синхронной вычислительной системы в том случае, если на все процессоры вычислительной системы удается подать один и тот же поток команд. Для этого могут быть использованы внешние контакты, осуществляющие прием запроса на обработку прерывания от внешнего управляющего устройства и ответ о готовности к обработке прерывания. В результате процессорная сеть переходит в состояние, когда на каждом процессоре выполняется своя подпрограмма обработки прерываний. После завершения которой процессорная сеть возвращается в прежнее состояние.

Классы алгоритмов шире, чем в трех ранее перечисленных случаях за счет гибкости придаваемый возможностями программирования транспьютеров и ТП ЦСП. Для выполнения операций обработки и коммуникации на разных процессорных элементах выделяются различные по длительности интервалы времени. Этот класс ВА на СБИС отличается моделью вычислений, которая позволяет одновременно реализовывать процессы на разных процессорах в режиме чистого физического параллелизма, а подпроцессы этих процессов могут быть выполнены псевдопараллельно на каждом процессоре. Основной режим – асинхронный режим работы.

Мультимикропроцессорные системы реализуют все арифметические операции, операции доступа к памяти всех предыдущих ВА, с другой стороны Т обладают большей памятью, в которую могут быть размещены как программы, так и данные. Одним из наиболее эффективным способом является управление потоками данных. Различие управления потоками команд от управления потоками данных: поток команд поступает в процессор, обрабатывает данные поступающие с устройства данных, как правило это программа записанная в памяти. В этом случае команда поступает в ЦП Т, инициирует выборку данных из памяти, обрабатывает их. Результат записывает в регистр или в память. В случае когда потоки данных управляют вычислениями в процессоре необходимо специальная логика работы процессорного элемента, которая заключается в том, что до тех пор пока не поступят все данные во всех входных потоках результат обработки данных не выдается в качестве выходного потока данных. Язык конкурентного программирования Occam (Occam 2), специально созданный для программирования транспьютеров имеет конструктор Alt и PRI\_ALT. Каждый из выполняемых процессов в этих конструкторах снабжается «сторожем»-процессом, который содержит условие, при котором, может выполняться каждый процесс. В качестве процессов могут быть процессы ввода и вывода данных. Таким образом, осуществляется альтернативный выбор каналов ввода-вывода данных. Отличия между этими конструкторами состоят в одном: в конструкторе Alt проверяется сначала условие «сторожа», и если это условие оказывается верным, то выполняется процесс. При этом может быть выполнен любой из списка процессов. При Alt осуществляет проверку условий строго начиная с первого процесса.

Для мультимикропроцессорной ВА характерным является то, что процессоры в каждый момент времени могут находиться в различных состояниях как в коммуникации, так и в фазе выполнения вычисления. С учётом использования принципа управления данными при вычислениях получается, что процессоры могут находиться ещё в одном состоянии, в состоянии ожидании данных. Для каждого ПЭ в итоге можно нарисовать ось времени, интервалы времени, которые использовались этим процессором для коммуникации, вычисления и ожидания. Таким образом, получается, что на эффективность вычислений мультимикропроцессорных ВА следует учитывать три фактора: время коммуникации, время ожидания, время вычисления. Наибольшая эффективность будет показываться теми системами, в которых суммарное время на ожидание и коммуникации при выполнении одной и той же задачи будет минимальным.

Выделяют 3 класса эффективных алгоритмов:

1. Класс алгоритмов с алгебраическим параллелизмом;

2. Класс алгоритмов с геометрическим параллелизмом;

3. Коллективного решения (параллелизм типа коллективного решения).

К каждому из этих классов можно отнести определенные алгоритмы с характерными признаками только для этого класса.

Алгебраический параллелизм выделяет из последовательного модульного алгоритма, в результате которой он делится на ряд вычислительных модулей. Алгоритм отображается на процессорную сеть. В результате чего вычислительные модули распределяются по процессорам, а каналы передачи данных тоже распределяются по внутренним и внешним физическим каналам передачи данных.

Геометрический параллелизм также выделяется из последовательного модульного алгоритма. В этом случае может быть найден вычислительный модуль, который многократно может быть использован в процессорах ПС. В результате получается распределенный параллельный алгоритм, который обрабатывает данные с помощью одинаковых вычислительных модулей, которые обрабатывают одновременно, каждый в отдельности, объемы данных меньших размеров.

Для параллелизма типа коллективных решений характерным является наличие одного модуля мастера и n-1 модуля работника. Главная задача мастера – распределение работ между работниками наилучшим образом с точки зрения загрузки их вычислительной работой.

Элементная база Т и ТПЦСП позволяет поддерживать 2 модели вычислений: псевдопараллельных вычислений на каждом процессоре и модель физического одновременного выполнения вычисления на многих процессорах. На каждом процессоре имеется аппаратный планировщик процессора, который осуществляет псевдопараллельное выполнение подпроцессов. Псевдопараллельное вычисление процессов сводится к тому, что каждый ПЭ в каждый момент времени выполняет только один процесс (часть этого процесса). В целом, физический параллелизм состоит в возможности одновременного выполнения минимум одной команды на каждом процессоре сети. Т.о. осуществляет выполнение одновременно многих процессов.

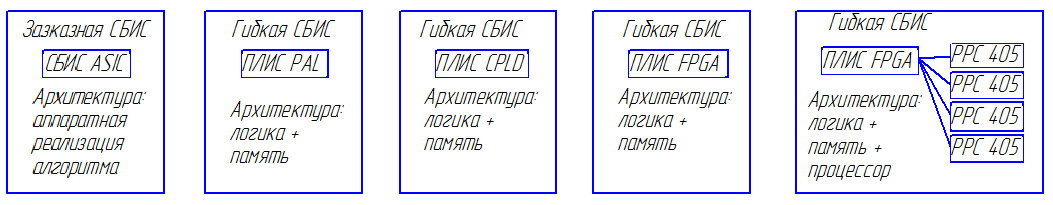
Нейровычислительная ВА (VI) наилучшим образом реализуется с помощью нейрочипов, в которых наиболее эффективно выполняются алгоритмы, приведенные к нейровычислительному виду. Каждый нейрочип как правило содержат сумматоры и умножители, а также устройства хранения весовых коэффициентов. Эти устройства поддерживают передачу данных между отдельными нейронами. Для реализации нейронных сетей могут быть использованы, кроме нейрочипов, систолические и медийные процессоры, универсальные процессоры из процессорного ряда Intel. Однако с помощью этих процессоров может быть произведена эмуляция нейронных сетей. Физическая реализация достигается на нейрочипах, каждый из которых содержит определенное количество нейроэлементов с определенным количеством входных и выходных каналов. Для реализации несложной нейросети, имеющей 1-2 слоя обработки, может потребоваться один или более нейрочипов. Реализация на основе универсальных процессоров Intel с использованием специализированных библиотек позволяет эмулировать нейронные сети, т.е. планировать выполнение нейроалгоритмов средствами одного процессора с помощью бесплатно предоставленных библиотек или бесплатных программных продуктов. В Matlab имеется библиотека Fuzzy logic для моделирования нечетких алгоритмов. Количество используемых нейрочипов при физической реализации должно соответствовать сложности нейроалгоритма, полученного из исходного путем эквивалентной трансформации. Для реализации нейроалгоритма с помощью нейрочипа может потребоваться не один нейрочип, поэтому количество используемых нейрочипов для физической реализации должно соответствовать сложности нейроалгоритма, полученного из исходного последовательного алгоритма путем трансформации в параллельный нейроалгоритм. Для реализации нейровычислительных ВА может быть использована современнаяя элементнаяя база ПЛИС, в которой могут быть созданы аппаратные функции, реализующие нейроны.

Универсальные ВА (V) используют универсальные МП и обладают возможностью реализации полного пространства алгоритмов, однако эти алгоритмы реализуются с различной эффективностью, т.к. они требуют различные вычислительные ресурсы (производительность, память, коммуникационные каналы). Как правило, реализации этих алгоритмов уступают реализации в специализированных ВА. Использование средств операционных систем, систем программирования (языков), программных продуктов, придает системам этого класса свойство универсальности и позволяет разрабатывать и использовать любые алгоритмы из полного пространства алгоритмов. Вследствие этого на первой стадии разработки аппаратно-программных комплексов (АПК) для специализированных ВА на СБИС могут быть использованы универсальные ВА на СБИС. Это делается для тестирования разрабатываемых специализированных ВА на СБИС путем сравнения с результатами работы алгоритмов универсальных ВА на СБИС.

С точки зрения возможности реализации операций универсальные ВА способны выполнить все операции уже названных специализированных ВА. ПЭ обладают большим набором команд для реализации самых различных операций. В этот класс ВА входят ПК, серверы, суперЭВМ, а также ранее выпускавшиеся ПК на транспьютерной элементной базе. Для управления универсальными ВА кроме аппаратных средств используются системные программные средства – операционные системы общего назначения и специального назначения, среди которых ОС реального времени, ОС, поддерживающие работу больших распределенных сетей микропроцессоров. Для автоматизации использования используются языки программирования, используются программные продукты инструментального и прикладного назначения.

В последнее время реализованы новые классы универсальных ЭВМ, такие как квантовые компьютеры и компьютеры обратимых вычислений: D-Wave – квантовый компьютор на 512 кубитах (2012 г.) и на 128 кубитах (2009 г.).

Куби́т (q-бит, кьюбит, кубит; от quantum bit) — квантовый разряд или наименьший элемент для хранения [информации](http://ru.wikipedia.org/wiki/%D0%9A%D0%B2%D0%B0%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C) в [квантовом компьютере](http://ru.wikipedia.org/wiki/%D0%9A%D0%B2%D0%B0%D0%BD%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80).

****

**Рис. 16 Заказная и гибкая СБИС-реализация алгоритма**

Одной из проблем реализации специализированных ВА на СБИС является необходимость получения из исходного алгоритма, который часто является последовательным, нового эквивалентного алгоритма предназначенного для реализации в специализированной ВА на СБИС, точно так же при реализации алгоритмов в универсальных ВА может возникнуть проблема связанная с некорректным выполнением алгоритмов и их программ через неопределённый промежуток времени. Для решения этой проблемы разрабатывается дополнительное программное обеспечение, позволяющее возвратить ПО в режим корректного выполнения, это связано с возникновением критических ситуаций при работе ПО.

При реализации алгоритмов, предназначенных для выполнения на специализированных ВА на СБИС, на универсальных ВА на СБИС разрабатывается дополнительное программное обеспечение, позволяющее эмулировать или спланировать выполнение специализированного алгоритма на универсальной системе, что в целом снижает эффективность работы по быстродействию, но позволяет получить важные данные по правильности работы системы. С другой стороны при реализации специализированных ВА на СБИС с параллельной обработкой команд и данных, возникают три критические ситуации. Возможность моделирования на универсальной ВА позволяет разработать дополнительно АО и ПО, которое нельзя было ранее предусмотреть при разработке специализированной системы, вследствие получения неправильных результатов.

Виды критических ситуаций:

* «смертельное объятие» - Deadlock (программа попадает в тупик, система прекращает работу)
* «живое объятие» - Lifelock (процессор работает, но получения конечных результатов нет или можно заметить, что программа работает в бесконечном цикле, не позволяющем получать результаты)
* «неопределенная отсрочка» - Undefined Postponement (некоторый программный модуль или модули на протяжении длительного времени не могут получить доступ к данным, и их выполнение постоянно откладывается на неопределенное время)

Для выявления критических ситуаций удобно использовать отладчик программного обеспечения, позволяющий провести анализ содержимого регистров центрального процесса. В частности указатели стека и регистра «счетчик адресов», в котором всегда содержится указатель на адрес следующей выполняемой команды, но существуют некоторые особенности этого процесса в случае наличия данных в ячейках памяти следующих сразу за командой

Критические ситуации проявляют себя не сразу и связаны с самопроизвольным изменением работы прикладного ПО, который ведет в целом к получению неправильных результатов. При этом поведение прикладного ПО можно классифицировать в виде этих критических ситуаций. Первый вид ведет к тому, что ПО прекращает выполняться, что подтверждается тем, что содержимое счетчика команд не изменяется, то есть не выбирается следующая команда программы. В случае возникновения второй критической ситуации реально происходит следующее: ПО продолжает выполняться, но попадает в такой программный цикл, из которого оно не может самостоятельно выйти. При этом содержимое счетчика команд постоянно изменяется, то есть выбираются команды из программы. При возникновении третьей ситуации некоторые части алгоритмического обеспечения, реализованные в программе, становятся недоступными для выполнения. В этом случае счетчик команд постоянно изменяется, то есть программа работает, при чем счетчик команд имеет широкий спектр изменения адресов. При тестировании такой критической ситуации становится понятным, что часть процессов не выполняется в следствие того, что другие программные процессы оказываются более конкурентными по сравнению с ними в борьбе за право быть выполненными ЦП. Причина этого может крыться например в программном интерфейсе, с помощью которого процессы получают доступ к одним и тем же данным. Процессы, имеющие быстрый интерфейс доступа к данным, несмотря на равный приоритет программ получают преимущество для выполнения. Процессы, имеющие сложный и более длительный процесс выполнения, уступают в доступе к данным для процессов, имеющих быстрый интерфейс (рис 17).

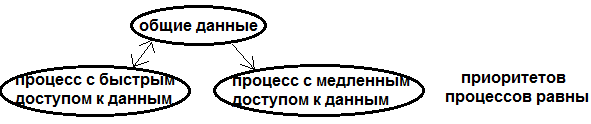


Рис. 17 Процесс с медленным доступом к данным может получить отказ при доступе к данным

Имеются 2 варианта работы алгоритма:

1. Пусть данные только читаются. Тогда проблемы не возникают.
2. Пусть процесс с быстрым доступом начинает менять общие данные. Тогда процесс с медленным доступом может остаться на «голодном пайке».

При разработке информационного АО и ПО несмотря на малую вероятность возможности возникновения каждой из этих критических ситуаций следует заранее предусмотреть в ТЗ на разработку ПО создание программ, контролирующих возникновение критических ситуаций и алгоритмов их нейтрализации. В противном случае можно столкнуться с неправильной работой прикладного ПО через непредсказуемые промежутки времени.

Для обнаружения и выхода из них требуется разработка дополнительного алгоритмического и аппаратного обеспечения, разрушающую критические ситуации.

При описании современных ВА на СБИС следует акцентировать внимание на элементную базу каждой из них и те операции, которые могут быть выполнены с помощью элементной базы в алгоритмах.

В ВА память не выполняются вычислительные операции, Логика + память выполняются операции доступа к памяти, логические, АА арифметические, коммуникационные и вычислительные операции, которые могут быть реализованы с помощью структур, находящихся на кристалле, а так же операции ввода-вывода данных в/из структур, находящихся на кристалле.

С помощью ПЛИС можно создать аппаратные вычислительные структуры для реализации алгоритмов, не содержащие избыточных вычислительных элементов. Систолическая архитектура: логические, арифметические операции. Для этих целей могут быть использованы процессоры, имеющие только регистры (без памяти), так и процессоры, имеющие регистры и память, в которых содержатся данные и программы. В первых трех ВА на СБИС алгоритмическое обеспечение реализуется аппаратно.

Мультимикропроцессорные ВА на СБИС позволяют реализовывать алгоритмы, содержащие различные операции из системы команд процессорных элементов. С помощью элементной базы создаются аппаратно-программные комплексы (АПК).

Нейровычислительная архитектура требует реализации операций доступа к памяти, операций сложения и умножения, передачи данных. Для этих целей используются нейрочипы.

Универсальная архитектура реализует без ограничений все операции, содержащиеся в алгоритмах средствами системы команд процессора. Для новейших квантовых компьютеров и компьютеров обратимых вычислений в последние годы создана соответствующая элементная база. Для супер ЭВМ, например китайского производства, была разработана в 2012 году новая процессорная элементная база, за счет этого они обогнали американцев.

## 1.5. Элементная база ВА на СБИС

Основные виды процессоров и вспомогательных элементов:

1. универсальные процессоры (МП) (Intel-, AMD-совместимые)
2. транспьютеры (Т) (Inmoos Ltd)
3. цифровые сигнальные процессоры (ЦСП), (Texas Instruments, Analog Devices, Motorola) ТПЦСП, разработанные для цифровой обработки сигналов и параллельной обработки данных, имеющие более высокую производительность по сравнению с МП, МК.
4. микроконтроллеры (МК) – универсальные микроконтроллеры, внутренняя архитектура и ресурсы которых ориентированы на реализацию алгоритмов управления
5. программируемые логические интегральные схемы (ПЛИС) для реализации реконфигурируемых аппаратных вычислительных систем. На кристалле ПЛИС имеется 5 групп элементов: матрица конфигурационно-логических блоков, конфигурационно-логические блоки (КЛБ), блоки синхронизации, блоки ввода-вывода, блочная память, в которую записываются данные и внутренняя конфигурация, Versa Ring – интерфяейс ввода-вывода с дополнительными трассировочными ресурсами по периферии кристалла, АЦП и ЦАП, параллельные и последовательные интерфейсы многих стандартов. ПЛИС конфигурируются в ВС с высокой степенью параллельной обработки команд и данных, и позволяет реализовывать синхронные и асинхронные аппаратные вычисления.
6. БИС и СБИС памяти

Интегральная память делиться на постоянную и оперативную.

Оперативная память делится на:

* Статического типа (транзисторная более быстрая)
* Динамического типа (конденсаторно-транзисторная, более медленная)

1. нейрочипы – используются для аппаратной реализации нейроалгоритмов. На их кристалле находятся большое количество нейроэлементов (нейроны), которые могут объединяться в слои, обрабатывающие входные и выходные данные с изменяемой разрядностью за счет одновременной работы большого количества элементов достигается очень большая скорость обработки информации.
2. адаптеры (А) – используются в качестве промежуточного звена в коммуникации каналов для преобразования параллельной байтовой передачи данных в последовательную байтовую передачу данных и наоборот. Они бывают нескольких типов и позволяют осуществлять преобразование последовательное программное и в режиме прерывания программы.
3. электронные коммутаторы (К) – используются в коммуникационных каналах для программного переключения каналов в мультиплексорах мультимикропроцессорных ВА на СБИС с квазистатической, динамической и статической архитектурой соединения ПЭ.

Статическая архитектура предполагает выполнение ПО от начала до конца без изменения конфигурации ПС.

Квазистатическая архитектура – для каждого интервала времени имеет место конкретная новая конфигурация соединения ПЭ (разное количество ПЭ и коммуникационных каналов) при этом на каждом интервале времени все вычисления должны быть завершены и получены промежуточные конечные результаты.

Динамическая архитектура создается с помощью специально выделенного ПЭ, который контролирует степень загруженности ПЭ в сети и программно изменяет внутреннюю конфигурацию сети между ПЭ с целю более полной загрузки их вычислительной работой.

1. интерфейсные схемы – могут быть применены для согласования форматов передаваемых данных и соответствующих электрических параметров (сигналов).
2. и другие:

Сопротивления, емкости, индуктивности и другие радио-элементы, которые размещаются на процессорных платах вместе с элементной базой ВА на СБИС, являющиеся пассивными элементами.

Для реализации ВА на СБИС достаточно перечисленной элементной базы.

ЭБ ВА на СБИС постоянно совершенствуется и часть компонент, которая раньше находилась вне кристалла ПЭ, например К, в настоящее время реализуется непосредственно на кристалле ЦСП, ТПЦСП, ПЛИС.

# 2. Особенности обработки информации в вычислительных архитектурах

Аппаратному обеспечению ставится в соответствие определенный класс алгоритмического обеспечения. Соответствие определяется как максимальная эффективностьи,с точки зрения выбранного критерия выполнения класса алгоритмов на классе аппаратного обеспечения.

## 2.1. Параллельная обработка информации в вычислительных архитектурах на СБИС

Параллельная выполнение команд и обработка данных в вычислительных архитектурах (ВА) на СБИС применяется в аппаратно-программных комплексах (АПК) реального времени вычислительной системы общего и специализированного назначения. Каждая из ВА на СБИС в табл. 1 обладает рядом особенностей. Но общими для них являются две:

1. только определенные алгоритмы могут быть наиболее эффективным образом реализованы в каждой из них;
2. при построении аппаратного обеспечения для каждой из них наилучшим образом может быть применена только определенная элементная база.

Эффективность реализации алгоритмов связана с использованием ВА на СБИС, а эффективность их реализации зависит от выбора элементной базы. Для эффективного применения в ВА на СБИС алгоритма актуальна разработка структуры сети процессоров, распределение на нее программных модулей.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблица 1. ВА на СБИС | | | |  |
| № | Название ВА | Реализация ВА | Алгоритмы ВА | Ускорение |
| 1 | Память | СБИС ОЗУ, ППЗУ | Доступ к памяти | 101 |
| 2 | Память + логика | ПЛИС типов: PAL, CPLD, FPGA | Реализация в функциональных устройствах реконфигурируемых аппаратных ВС (команд, функций, подпрограмм) в виде процессоров | 101-102 -107  f(L,V,ak,Par,Seq,Vкп)\*  Крп - количество блоков распределенной памяти |
| Память+логика+ процессор | ПЛИС типа: FPGA с 4 ядрами PowerPC405, PowerPC405 D5 |
| 3 | Систолическая | СП, ПСП ЦСП, ТП ЦСП | Систолические | 101-103 |
| 4 | Мультимикропроцессорная | Транспьютеры, ТП ЦСП | Классы эфф. алгоритмов алгебраического, геометрического, коллективного параллелизма | N раз, где N – количество ПЭ |
| 5 | Универсальная | Универсальные МП фирм Intel, AMD, VIA, др., транспьютеры, ЦСП | Алгоритмы последов. параллельной обработка команд и данных, планирования вычислений | f(n,V,k)\*\*, где n – количество процессоров, V – ёмкость ОЗУ, k – пропускная способность каналов передачи данных, простейший вариант – линейная целевая функция  u=k1\*n+k2\*V+k3\*K |
| 6 | Нейровычислительная | Нейрочипы, ТП ЦСП, транспьютеры, универсальные МП фирмы Intel | Нейроалгоритмы(де-терминированные и недетерминированные) | 10n, где n – число, имеющее некоторое ограничение. Единицами оценки производительности явл-ся кол-во соединений в секунду между отдельными нейронами и время , необходомое для проведения обучения. |

Для умножения двух матриц имеем u=tпосл/tпар= n3\*(t\*+t+)/(1\*t\*+n\*t+ )

\*f(L,V,ak,Par,Seq,Kрп) – зависимость для ускорения в вычислениях, где (1)L – количество логических переменных, (2)V – ёмкость ОЗУ на кристалле, ак – количество АЦП, ЦАП на кристалле, Раr – пропускная способность параллельного байтового интерфейса, Крп – количество блоков распределенной памяти. Разработчик используя инструментальные средства имеет возможность конфигурировать аппаратные возможности ПЛИС с целью достижения требуемых показателей эффективности.(по ускорению вычисления). При реализации на одном чипе ПЛИС достигается 107 (оценка сверху). Узким местом является передача данных между несколькими ПЛИС. При использовании двух и более ПЛИС узким местом является передача данных между ними при реализации сложных алгоритмов.

ПЛИС различных типов можно сравнивать по количеству на кристалле однотипных элементов.

\*\*Приведенная линейная целевая функция содержит весовые коэффициенты к1, к2, к3, но при этом это грубая оценка, которая учитывает наличие только одинаковых ПЭ, одинаковых емкостей памяти и каналов одинаковой пропускной способности.

Для реализации ВА используются СБИС ОЗУ (оперативное запоминающее устройство), программируемые логические интегральные схемы (ПЛИС), систолические процессоры (СП), цифровые сигнальные процессоры (ЦСП), транспьютеры, транспьютероподобные ЦСП (ТП ЦСП), универсальные микропроцессоры (МП) фирм Intel, AMD, VIA, др., нейрочипы.

Эти ВА являются специализированными и, за исключением универсальных ВА, реализуют узкие классы алгоритмов. Большей гибкостью обладают мультипроцессорные ВА из реконфигурируемых сетей программируемых процессоров. В систолической и нейровычислительной ВА выполняются только алгоритмы, преобразованные соответственно в систолические и нейровычислительные алгоритмы, эквивалентные исходным. ВА «логика+память» позволяет реализовывать новые команды, функции и алгоритмы обработки данных с высокой степенью параллелизма. В универсальной вычислительной архитектуре могут выполняться произвольные алгоритмы обработки и планирования обработки данных. Между эффективностью и универсальностью использования ВА на СБИС имеет место обратная зависимость.

### Разработка АПК

При проектировании алгоритмического и программного обеспечения необходимо разрабатывать соответствующую структуру аппаратного обеспечения, которая обеспечит при реализации использование особенностей и преимуществ алгоритмического программного обеспечения.

В настоящее время существует три подхода к разработке АПК:

1. традиционный, под которым следует понимать подход «от вычислительной системы (ВС) к алгоритму и далее к программе», т.е. подход «от платы к алгоритму» (под платой здесь подразумевается плата ВС с одним или несколькими процессорами);
2. подход «от алгоритма к ВС и далее к программе», т.е. подход «от алгоритма к плате»;
3. смешанный (комбинированный) – используются элементы первого и второго подходов.

Развитие этих подходов связано с требованиями:

* реализация подхода в масштабе реального времени;
* недостаточные вычислительные ресурсы однопроцессорной ВС (процессор, память, каналы);
* необходимость распараллеливания последовательных алгоритмов;
* разработка принципов параллельной обработки данных (1959);
* появление надежной элементной базы БИС и СБИС для реализации вычислительных систем (вторая половина 70-х – первая половина 80-х гг.);
* разработка ВА на СБИС.

Первый подход, «от платы к алгоритму»

Является традиционным, поскольку ориентируется на универсальные однопроцессорные средства вычислительной техники (например, персональные ЭВМ (ПЭВМ)) и используется, как правило, при разработке последовательных алгоритмов. При этом опираются на известную конфигурацию аппаратного обеспечения, операционную систему, языки программирования и т.д. Другими словами, известен тип процессора, его система команд, количество параметров, объём оперативной памяти, количество и тип каналов связи. На основе имеющихся данных формируется исходная постановка математической, информационной и других задач, разрабатывается алгоритмическое, а затем и программное обеспечение (ПО). Заданы процессорная сеть и алгоритм обработки. Требуется составить эквивалентный параллельный алгоритм, распределяющий программные модули по процессорам, а программные каналы – по аппаратным внешним и внутренним каналам, то есть задействующий все заданные аппаратные средства.

Возможна разработка параллельного алгоритмического ПО для однопроцессорной системы с использованием принципов разделения времени и различных приоритетов для модулей алгоритма и ПО.

При таком подходе большое значение имеют характеристики ВС, в частности, количество процессорных элементов, количество информационных каналов (внешних и внутренних), емкость оперативной памяти. Причём для многих современных систем наиболее важна возможность параллельного выполнения нескольких программных модулей (ПМ). Естественно, на одном процессоре достаточно сложно достичь одновременного выполнения двух различных программ, поэтому необходимо разделение по приоритетам и квантование по времени (для выполнения ПО).

|  |
| --- |
| Рис.24 . Пример подхода «от платы к алгоритму»  рис1  МПИ  ПЭ1  1-2  1-3  2-3  2-4  4-5  5-7  5-6 |

1. Для каждого процессора указывается тип процессорного элемента. Это является указанием для компилятора имеет ли процессор ПЭ ЛОЗУ, ОЛЗУ или Без памяти, а так же ППТ, разрядность данных и другие свойства.
2. процессоры: согласно данному подходу (рис. 7):

* на ПЭ1 реализуются ПМ1, ПМ2 и ПМ3;
* на ПЭ2 реализуется только ПМ4;
* на ПЭ3 реализуются ПМ5 и ПМ6;
* наконец, на ПЭ4 реализуется только ПМ7.

1. каналы: согласно данному подходу (рис. 7):

* каналы 1-2, 1-3 и 2-3 являются внутренними по отношению к ПЭ1;
* канал 2-4 является внешним по отношению к ПЭ1 и ПЭ2;
* канал 4-5 является внешним по отношению к ПЭ2 и ПЭ3;
* канал 5-6 является внутренним по отношению к ПЭ3;
* канал 5-7 является внешним по отношению к ПЭ3 и ПЭ4.

Процессор, не имеющий внутренней памяти (например, ПЭ2 на рис. 7), выполняет операции с помощью внутренних регистров:

* регистр А → первый операнд;
* регистр В → второй операнд;
* регистр С → операнд-результат.

ООЗУ – двухпортовая память, то есть одни и те же данные могут считываться двумя процессорами. Однако при этом необходимо ПО, блокирующее одновременную запись и считывание. Наряду с этим программным обеспечением может быть использовано аппаратное обеспечение с теми же функциями, иначе может возникнуть ситуация, когда будет считаться, что считана часть новой и часть старой информации, а результат обработки будет корректным. Т.е. алгоритмы обработки остаются правильными, а использование исходных данных будет относиться к двум разным этапам обработки, например к текущему и будущему этапу.

Итогом разработки с помощью традиционного подхода должно являться создание таблиц соответствий:

* Процессорный Элемент – Программный Модуль;
* аппаратные каналы и программные каналы (физические и логические).

При разработке алгоритмического обеспечения используются свойства аппаратного обеспечения, такие как количество процессорных элементов, количество внешних каналов, организация ОЗУ (локальные и общие). В зависимости от этого определяют, какие модули могут быть размещены на каждом из этих устройств.

Вывод. Традиционный подход позволяет перенести свойства аппаратного обеспечения на искомое алгоритмическое и программное обеспечение, т.е. разработать алгоритм и программного обеспечения на основе известной Вычислительной Системы.

Замечание. Цель разработки АПК определяется критерием эффективности и может быть сформулирована как достижений максимально возможного ускорения на заданной процессорной структуре. В результате разработки может быть получено несколько вариантов реализаций АПК. Следует выбрать наилучший из них (максимально ускоряющий процесс вычисления).

Второй подход, «от алгоритма к плате»

Возник позднее (в 80-е годы) и потому является более новым, современным по отношению к традиционному подходу. Он используется при разработке параллельных и псевдопараллельных АПК. При этом на основе известной постановки задачи (математической или информационной) необходимо разработать сначала структуру алгоритмического обеспечения, а затем для него разработать конфигурацию аппаратного обеспечения.

Под этим понимают:

* определение общего количества ПЭ;
* определение типа их соединения (внешние (соединяют ПЭ) и внутренние каналы (внутренние связи в ПЭ) для каждого ПЭ).
* определение емкости и типа оперативной памяти (ЛОЗУ, ООЗУ, без памяти).

Поскольку сначала известна математическая или информационная постановка задачи, этапы разработки алгоритма делятся на следующие шаги:

1. исходным является последовательный корректно работающий алгоритм (псевдопараллельный);
2. на его основании разрабатывается эквивалентный параллельный алгоритм (в смысле получения тех же результатов); Сущетвует несколько классов параллельных алгоритмов, соответствующих ВА на СБИС.
3. определяются свойства параллельного алгоритма, на основании которых разрабатывается аппаратное обеспечение. Сущетвует несколько подклассов в каждом классе параллельных алгоритмов, уточняющих свойства эффективных алгоритмов.

Замеры времени ложатся в основу различных полученных оценок эффективности, и на их основе выбирается лучший алгоритм и лучшая конфигурация АПК.

|  |
| --- |
| Рис. 25. Пример подхода «от алгоритма к плате» (отражение алгоритма на АПК)  рис2 |

При отображении свойств алгоритмического обеспечения на процессорную структуру может быть получено несколько вариантов реализаций аппаратного обеспечения, в котором будет выполняться конкретный вариант алгоритмического обеспечения. В этом случае следует произвести сравнительный анализ времени выполнения данного параллельного алгоритма в каждой процессорной структуре и выбрать лучший для реализации. Таким образом, мы пришли к выводу, что данное преобразование не является тождественно обратным.

Свойства параллельного алгоритма, на основании которых можно было бы разработать параллельную структуру процессоров *(основные 2-4)*:

1. модульность алгоритма - возможность одновременного функционирования в псевдопараллельном или реальном параллельном режиме (при разработке АПК опираются на модульную структуру параллельного алгоритма, который должен быть эквивалентен последовательному алгоритму в плане получения тех же результатов на тех же данных); для каждого модуля должна быть определена вычислительная или информационная процедура обработки данных; принципиально все ПМ могут быть разделены на две группы – модули обработки данных и модули передачи данных;

Свойство модульности алгоритма оказывает влияние на количество процессорных элементов (ПЭ) и, в целом, на конфигурацию процессорной структуры.

1. размер вычислительного модуля алгоритма, определяемый количеством операций, либо объемом вычислительного модуля, в состав которого входят данные. Если количество модулей в алгоритме влияет на количество процессорных элементов в вычислительной системе, то размер вычислительного модуля влияет и на емкость памяти.

При определении размера вычислительного модуля производится подсчет элементарных арифметических операций (к ним относят сложение «+», вычитание «-», умножение «\*» и деление «/»), причём непосредственный подсчёт размера ведётся операциями «+», «\*» и операцией «пересылки данных» (запись, считывание);

Свойство размера вычислительного модуля алгоритма, определяет объем локальной или общей памяти (ЛОЗУ, ООЗУ) процессорных элементов (ПЭ).

1. количество и тип каналов связи, коммуникационная структура каналов передачи данных: очевидно, что данные между вычислительными модулями алгоритма могут быть переданы посредством внутренних или внешних каналов связи. Внутренний канал реализуется в виде ячейки памяти, поэтому он считается абсолютно надёжным (если память исправна, то и канал тоже исправен). А для осуществления внешнего канала необходимо физическое соединение двух внешних интерфейсов двух различных процессорных элементов (ПЭ), поэтому внешний канал не считается абсолютно надёжным.

Способы управления вычислениями:

* Синхронный способ – предполагает выполнение для каждого конкретного интервала времени на каждом процессоре должны выполняться одинаковые операции или один и тот же программный модуль.
* Асинхронный способ – для каждого процессора задаются различные интервалы времени, в которые могут выполняться различные программные или алгоритмические модули. Для каждого программного модуля определяется процент его выполнения, после чего его выполнение приостанавливается для обмена данными с другими программными модулями.

Третий подход является смешанным (комбинированным)

Поскольку разработка искомого аппаратно-программного обеспечения вначале ведётся на основе первого подхода (рис. 7), а затем – на основе второго (рис. 8). Другими словами, разрабатывают некую версию вычислительная система (аппаратное обеспечение) и на её основе строят алгоритм, но при дальнейшем рассмотрении выявляют новые свойства и возвращаются обратно к аппаратному обеспечению, чтобы исправить (доработать) систему.

|  |
| --- |
| Рис. 31. Характер зависимости быстродействия от свойств алгоритма  D:\My Files\Documents\Мои Документы\Ботва\#ИУ1\ПАиПО [8-9 сем]\8 сем\2009\add\draw_14a.jpg |

То есть, правильно разделяя алгоритм на число модулей можно получить большее ускорение вычислительного процесса.

Вычислительный ресурс процессорной сети по производительности должен быть использован наилучшим образом. Для этого требуется, чтобы как можно большее время процессорные элементы выполняли полезную вычислительную работу по выполнению модуля алгоритма. Время передачи данных и время ожидания данных между процессорными элементами (ПЭ) и вычислительным модулями (ВМ) было минимальным. Метод решения задачи – балансировка загруженности вычислительной работой процессорных элементов.



Модульный алгоритм задачи должен быть распределен по процессорам таким образом, чтобы загруженность каждого ПЭ была как можно выше. Тогда все будет выполняться быстрее. С другой стороны, следует эффективно использовать коммуникационный ресурс процессорной сети. Это означает сокращение времени передачи данных и времени ожидания, данных процессорами.

Использование метода «аппарат-алгоритм» позволило достичь некоторого роста эффективности. Дополнительного увеличения эффективности работы можно достичь добавлением процессорного элемента.

|  |
| --- |
| Рис. 32. Характер зависимости ускорения от количества ПЭ    D:\My Files\Documents\Мои Документы\Ботва\#ИУ1\ПАиПО [8-9 сем]\8 сем\2009\add\draw_15b.jpg |

Из графика видно, что ускорение на пяти и шести процессорных элементов выше, чем на четырех процессорных элементов. Поэтому целесообразно разделить модули 3 и 4 на 3 модуля и загрузить ими 3,4 и 5 процессорные элементы. Общее время выполнения алгоритма сократится, а ускорение увеличится.

Время вычисления определяется по формуле

, где *N*, *K* и *M* – количество операций сложения, умножения и присвоения соответственно.

Оценка времени выполнения алгоритма

, где *S* - количество специфических операций. Под специфическими операциями понимают те операции, которые в основном используются в данном алгоритме (арифметические, логические, сравнение и др.).

Вывод. Используя смешанный подход, можно определить наилучшее разбиение для алгоритма на модули или наилучшее число процессоров для данной задачи, то есть когда будет достигнута требуемая эффективность АПК.

На характер зависимости оказывают влияние конкретные свойства, как алгоритма, так и аппаратного обеспечения

При передаче данных блоками затраченное время будет меньше, чем при передаче того же количества данных отдельными пакетами. Причина в использовании одной операции передачи всех данных и дешифрации только одного кода операции.

При переносе свойств алгоритмического обеспечения на аппаратное обеспечение будет достигаться наилучшая эффективность аппаратно-программного комплекса.

Перенос свойств аппаратного обеспечения в область алгоритмического обеспечения также приведет к созданию эффективных алгоритмов.

Третий подход позволяет разработать оптимальный АПК, обладающий наилучшими характеристиками и высокой производительностью в результате пошагового применения первого и второго подходов.

## 2.3. Критерии эффективности

Существует целый ряд критериев эффективности, которые позволяют оценивать эффективноссть реализации алгоритмического, програмного и аппаратного обеспечения

* Минимальное время выполнения (абсолютный критерий)

При оценке времени алгоритма в распоряжении могут быть использованы инструменты по подсчету характерных для алгоритмов операций. При оценках времени выполнения программного обеспечения используются другие инструменты, связанные с замерами времени выполнения программы. Может быть оценено по количеству используемых операций, как то: +, –, другие элементарные арифметические операции, операции передачи данных. Также для оценки используют умножение с накоплением  (актуально для работы с матрицами).

Машинные команды имеют разную длительность вследствие различных способов адресации операндов. Если в качестве операндов команда использует данные регистров, операция будет выполняться быстрее всего, так как данные уже загружены в регистры другими командами. Иные способы адресации требуют дополнительных рабочих циклов обработки команд.

# MOV R1,R2 – регистровый способ адресации

# MOV (R1)+,(R2)+ автоинкрементный способ

# MOV -(R1),-(R2) автодекрементный способ

Процессор осуществляет проверку правильнотси команд вцелом

Поэтому подсчет количества операций дает оценку времени выполнения алгоритма (но не эффективности работы программного обеспечения и вычислительной системы). Время выполнения каждой команды можно вычислить по формуле:



Вывод: Оценка скорости выполнения алгоритма может быть дана в количестве операций в единицу времени. Если для каждой операции дана скорость ее выполнения, то оценка производится в единицах времени (например, в секундах).

Особенности технической реализации аппаратного комплекса могут вносить изменения во время выполнения алгоритма. Для составления характеристики выбирают программу, выполняемую за минимальное время.

По методу подсчета характерных операций погрешность расчета времени выполнения алгоритма по этому методу составляет до ~10%, что является весьма грубой оценкой.

Наиболее точная оценка времени выполнения алгоритма производится с помощью специальных таймерных программ.

* Максимальное ускорение вычислений (относительный критерий)

Данный метод не принимает во внимание наличие других операций кроме учитываемых операций. А также использует операции с целочисленными данными и с плавающей точкой, обычной и двойной точности, различными видами адресации операндов.

Следует так же учитывать особенности выполнения команд в разных семействах процессорных элементов.

Мы всегда предполагаем, что время выполнения операции с плавающей точкой больше времени выполнения операции с фиксированной точкой. Например, на ЦСП, наоборот, вследствие аппаратной реализации аппаратного умножителя, что и приводит к более быстрому выполнению операции с плавающей точкой, время выполнения операций с плавающей точкой заметно меньше по сравнению с целочисленными данными (до 2-х раз меньше)

Для ЦСП ADSP2106 характеристика производительности 80 mips и 160 hflops

Ускорение вычислений (U) — величина безразмерная. Оценивается время выполнения базового алгоритма в сравнении с его новой версией:



Сравнение обычно производят для строго последовательного алгоритма и его реализации на одном процессоре. Затем разрабатывается параллельный алгоритм, эквивалентнй в список полученных тех же значений, что и в последовательном алгоритме. Затем производят подсчет коолличества операций до начала работы алгоритма до получения конечного результата. И, наконец, составляется соотношение для вычисления ускорения.

Если U > 1? То полученный параллельный алгоритм является более эффектиынм по сравнению с исходным последовательным.

Можно сравнивать 2 версии последовательных алгоритмов, так же можно рассматривать целый ряд параллельных алгоритмов. Наибольшей эффеквтиновстью будет обладать тот параллельный алгоритм, для которого будет получено максимальное значение ускорения вычислений.

В общем случае сравнивать можно две любые реализации на различном количестве процессорных элементов. Т.е. можно сравнивать между собой параллельныйе алгоритмы , однако при таком сравнении можно упустить очень важный момент, связанный с возможностью более быстрым выполнением последовательного алгоритма по сравнению с параллельным алгоритмом.

Наиболее точные оценки можно получить для ускорения при использовании замеров времени.

Также учитываются другие критерии, позволяющие принять качественное решение в отношении АПК. Например, так называемые оптимистические и пессимистические критерии в случаях, когда требуется проверка и переработка полученных результатов.

* Качественная оценка ускорения вычислений (качественный критерий)

|  |
| --- |
| рис45  Рис. 33. Оценки ускорения вычисления |

На плоскости (YN) проводятся кривые соответствующие функциям указанным ниже, которые делят первый квадрант на четыре области.



Если:

* Рассчитанное У попало в область недостижимого ускорения → в полученном алгоритме или замерах времени имеется ошибка, т.к. ресурс по производительности ограничен сверху по количеству процессорных элементов.
* Рассчитанное У попало в область пессимистичных ускорений → результаты проектирования системы вполне могут быть улучшены (по ускорению);
* Рассчитанное У попало в область оптимистичных ускорений → улучшить результаты скорее всего уже не удастся;
* Рассчитанное У попало в область сомнительных ускорений → результаты необходимо тщательно проверить и тем самым найти допущенную где-то ошибку.
* Стоимость АПК (экономический критерий)

Стоимость разработки АПК можно оценивать различными методами:

* без учета стоимости этапов разработки. В этом случае стоимость в основном зависит от количества ПЭ, емкости памяти, количества каналов данных, цены материнской и процессорной плат;
* с учетом этапов разработки программного обеспечения. Следует заметить, что в этом случае создание алгоритмического, информационного, программного и аппаратного обеспечения состоит из пяти стадий: техническое задание, эскизный проект, технический проект, рабочий проект, внедрение. Затраты связаны с оплатой труда сотрудников, созданием макетов, стендов и т.п. Стоимость разработки программного обеспечения обычно находится в пределах 60% от всех затрат. На стоимость влияют: процессорная плата, количество процессорных элементов, объем ОЗУ, количество каналов передачи данных и управляющих сигналов.
* Стоимость выполнения одной операции (относительный критерий стоимости)



Данный критерий показывает, что стоимость выполнения одной операции в дорогостоящей высокопроизводительной вычислительной системы может быть заметно ниже по сравнению со стоимостью выполнения одной операции в низко-производительной, но относительно дешевой вычислительной системы.

Эта характеристика имеет значение для вычислительных систем реального времени.

Реальная производительность определяется тестированием с использованием замеров времени выполнения конкретной команды.

* Комплексная оценка эффективности использования процессорного элемента (комплексный критерий)

Эффективность применения процессорных элементов можно оценить с помощью критерия, в котором учитываются технология реализации процессорного элемента, количество БИС или СБИС, необходимые для реализации вычислительной системы, производительность и энергопотребление.

Данный критерий сокращенно называется FOM.



Существуют также и другие критерии эффективности, позволяющие оценивать эффективность реализации АПК.

## 2.4. Ресурсы вычислительной системы

Обычно ресурсы вычислительной системы определяются тремя характеристиками.

1. Производительность вычислительной системы
2. Емкость памяти (в частности ОЗУ)
3. Пропускная способность каналов связи

### Процессоры

Система из одинаковых процессорных элементов называется однородной, из разных элементов — неоднородной. Чем больше процессорных элементов в однородной системе, тем выше ее производительность. Эта зависимость прямая, но не линейная. С одной стороны использование более мощного процессорного элемента в «узком месте» (канал передачи данных между процессором и памятью) позволит увеличить производительность всей системы. Но с другой стороны, при увеличении числа процессорных элементов имеются потери на синхронизацию процессов (процессорные элементы могут работать с разной скоростью), возникают потери на ожидание данных.

Пропускную способность можно увеличить, например, введением отдельного канала для передачи команд между памятью и процессором, канала для передачи данных между процессорными элементами и внешним устройством.

|  |
| --- |
| Рис.34. Пропускная способность канала («бутылка перевернутая») |

«Узкое место» по пропускной способности канала данных ограничивает производительность системы. Это связано с задержками передачи данных, так как медленнее всего действует канал передачи данных, через канал между памятью и процессором осуществляется ввод и вывод данных.

«Узкое место» по пропускной способности каналов данных ограничивает производительность системы. Примером расширения является процессорная сеть (рис.18).

|  |
| --- |
| ПЭ  ПЭ  ПЭ  ПЭ  Рис. 35. Расширение «узкого места» по производительности |

Большее количество процессорных элементов и каналов передачи данных между ними решает проблему «узкого места» по пропускной способности каналов данных между памятью и процессором.

Проблема «узкого места» по производительности вычислительной системы связана с ограниченными возможностями по производительности одного процессора. Недостаточная емкость или несоответствующая организация памяти также являются ограничениями для общей производительности. У каждого процессорного элемента может быть локальная либо общая память.

В общем случае, для однородной и неоднородной вычислительной системы, неравномерная загруженность процессорных элементов вычислительной работой и различные объемы передачи данных между процессорами оказывают существенное влияние на реально достигаемую производительность системы.

Балансировка загруженности многопроцессорных вычислительных систем позволяет повысить реальную производительность и приблизиться к максимально возможной. Балансировка связана с непроизводительными затратами времени на ожидание получения данных от соседних процессорных элементов.

Вывод: Ресурс производительности потенциальной и реальной системы зависит от количества процессорных элементов в однородной и неоднородной системе.

Если программа выполняется на различных ресурсах вычислительной системы, то это свойство называется масштабируемостью. Чтобы ускорить выполнение последовательного программного обеспечения, можно применять суперскалярные процессоры (суперскалярность — способность в одном такте вычислений вырабатывать сразу несколько скалярных результатов): в суперскалярном процессоре ускорение производится за счёт одновременной работы большого числа внутренних устройств.

Проблема расширения «узкого места» решается на двух уровнях:

* На верхнем уровне решается за счет возможности использования нескольких процессорных элементов, которые легко объединяются в сложные процессорные устройства с помощью внешних коммуникационных «линков» или общей ОЗУ.
* На нижнем уровне эта проблема может быть решена путем использования процессоров имеющих различную внутреннюю архитектуру (RISC и CISC) позволяющих с различной эффективностью выполнять простые и сложные команды.

В настоящее время существуют процессоры с сокращенным и полным набором команд (RISC и CISC). Каждый из этих классов процессоров эффективен в своей области применения.

# 3. Способы реализации вычислительных архитектур

## 3.1. RISC– и CISC–процессоры

RISC — reduced instruction set computer. CISC — complete instruction set computer.

RISC-процессоры имеют систему команд, состоящую из достаточно простых команд, осуществляющих:

* обмен данными между памятью и регистрами центрального процессора
* обмен данными между регистрами центрального процессора и памятью
* обработку данных в регистрах.

CISC-процессоры имеют в системе команд, сложные команды, позволяющие с помощью одной команды:

* обрабатывать векторные данные (векторных операндов)
* получать скалярные и векторные значения

RISC — архитектура процессоров, построенная на основе сокращенного набора команд (содержит часто в три раза меньше, чем у аналогичных по разрядности обычных CISC-процессоров, имеющих как сложные, так и простые команды).

RISC-архитектура характеризуется:

* наличием команд фиксированной длины
* большим количеством регистров
* операциями типа регистр-регистр (все, кроме команд чтения и записи в память)
* отсутствием косвенной адресации.

Одним их недостатков RISC-архитектуры считается фиксированная длина команд, требующая для хранения программы большего объема памяти. С другой стороны, она сильно упрощает конвейеризацию - конвейерную обработку команд и данных.

В процессе развития вычислительных систем возникала необходимость введения сложных команд, соответствующих решаемым задачам. Это помогло уменьшить объем двоичного кода программ и, как следствие, минимизировать время выполнения программ.

Команды «регистр-память» и «память-регистр» нужны для разгрузки и загрузки рабочих регистров.

### Типы команд

Команды делятся на простые и сложные команды.

* простые

<Операнд 1, Операнд 2→ Результат>

<регистр, регистр → регистр>

Бинарные операции

< память, память → память>

<Операнд → Результат>

Унарные операции загрузки регистров и записи в память

< память → регистр >

< регистр → память >

Имеются бинарные арифметические (+,-,\*,/), логические (AND, OR, XOR, NOT) и другие унарные операции.

В ЦП используется ограниченный набор регистров общего назначения, доступных для пользователя. Для выполнения операций над данными, которые хранятся в памяти сначала требуется использование команды типа <память → регистр>, затем еще одной такой же команды, если это двухместная операция, после этого команда <регистр, регистр → регистр> и обязательно надо выгрузить результат из регистра в память командой <регистр → память>. Т.о. для выполнения одной операции потребуется выполнение четырех простых команд. В случае, если результат операции может быть оставлен в регистре для выполнения одной операции потребуется меньшее количество команд загрузки и выгрузки данных, поэтому в архитектуре RISC-процессоров стремятся использовать оптимальное количество регистров.

При выполнении простых команд наблюдается следующее:

* они выполняются быстро
* требуется большое количество команд для загрузки и выгрузки данных, так как для выполнения команды приходится многократно осуществлять две фазы:
* выборка команды
* дешифрация команды

Виды адресации операнда:

* Регистровая (R)
* сложные

<Операнд 1, …, Операнд n→ Результат 1, …, Результат k>

<Регистр 1, …, Регистр n→ Регистр 1, …, Регистр k>

Сложные операторы модифицируют одной операцией содержимое групп регистров и содержание ячеек памяти. Для их реализации применяется микропрограммирование, чтобы сократить количество устройств, размещенных на кристалле. Примером сложной команды является команда «умножение с накоплением» c=a\*b+c. Для её выполнения необходимо предварительно с помощью простых команд загрузить из памяти регистры, а после выполнения выгрузить данные в память. Другим типичным примером сложных команд является выполнение операции скалярного умножения двух векторов. В системе команд получили распространение так же и команды с тремя операндами.

Виды адресации операнда:

* Регистровая R
* Прямая (R)
* Автоинкрементная (R)+
* Автодекрементная -(R)
* Индексная X(R)
* Относительная (счетчик команд) X(CK)
* Абсолютная #100
* @#100

Также команды можно разделить:

* скалярные: входные операнды и результат являются числами
* векторные: входные операнды и возможный результат являются векторами или массивами чисел. При этом для преобразования данных используется одна векторная команда. Результат векторной команды, как правило, может быть получен при выполнении набора скалярных команд.

При использовании векторных команд ускоряется обработка данных за счет исключения затрат времени на выборку и дешифрацию команд обработки одинаковых для всех компонент входных массивов.

### Фазы обработки команд

Различные процессоры используют различное количество фаз обработки. Рассмотрим основные фазы обработки одноместной команды.

|  |
| --- |
| Рис. 41. Фазы обработки команд |

Приведенные фазы обработки могут быть разбиты на подфазы, например, при выполнении дешифровки можно выделить подфазы дешифрации кода операции и дешифрации типа операнда. Известны процессоры с количеством фаз обработки от трех до двадцати и более. Фазы обработки команд могут быть использованы для создания конвейера команд центрального процессора. При этом конвейер повышает скорость обработки и выполнения приблизительно в прямо-пропорциональной зависимости от количества фаз обработки команд. Для эффективной работы конвейер команд должен быть сбалансирован, т.е. время обработки на каждой стадии должно быть одинаковым.

Однако использование векторных команд требует дополнительной подготовки векторизированного кода программы или разработки параллельного кода программы.

Повышение производительности микропроцессоров (МП) связано с:

* увеличением тактовой рабочей частоты (для этого необходимо постоянное совершенствование технологии производства СБИС, связанное с уменьшением технологических норм производства)
* использованием в процессорном ядре процессорного элемента нескольких процессоров
* ADSP2116 – два процессорных ядра
* TMS320C8\* - 3 или 5 ядер (среди которых всегда один с плавающей арифметикой, остальные обладают только целочисленной арифметикой)
* 2-х и 4-х ядерные процессоры AMD и Intel (в конце 2009 года планируется выпуск 6 и 8 ядерных процессоров)
* IMB CELL – 9 ядер
* усовершенствованием параллельной и конвейерной архитектуры обработки данных
* уменьшением времени доступа к памяти или использованием большого числа регистров
* увеличением пропускной способности и количества каналов при передаче команд и данных
* использованием в ЦП внутреннего конвейера обработки команд

|  |
| --- |
| Рис. 42. Внутренний конвейер обработки команд |

Т.к. здесь конвейер состоит из 3-х ступеней, то фазы обработки команд должны быть сгруппированы в 3 группы. На первую стадию подаются команды, на вторую – данные. На последней стадии обработки происходит запись результата.

|  |
| --- |
| Рис. 43. Внешний конвейер обработки данных |

Современные МП имеют 4-10 и более обрабатывающих устройств, представляющих собой конвейер. На нижнем уровне используется конвейер команд ЦП, производящий обработку команд. На верхнем уровне сами процессоры могут образовывать конвейер. Каждый такт обработки на каждой ступени конвейера должен составлять примерно одинаковое время. В этом случае конвейер является сбалансированным и работает эффективно.

Исключение составляют:

* начало обработки
* завершение обработки
* подача результата на начало конвейера

В этом случае не все стадии конвейера будут использоваться.

Так же для повышения эффективности можно использовать в ЦП несколько внутренних конвейеров команд.

|  |
| --- |
| Рис. 44. Три параллельных внутренних конвейера обработки команд |

|  |
| --- |
| Рис. 45. Внешний конвейер с использованием процессорных элементов с несколькими внутренними конвейерами |

Внутри ПЭ может быть несколько внутренних конвейеров обработки команд.

Например, в процессоре Itanium используются три конвейера команд, для реализации концепции явного параллелизма в алгоритмах EPIC. На этапе компиляции выделяются ветви программы, которые могут исполняться параллельно. Они будут выполняться на разных конвейерах команд. Код команд, соответствующих параллельным ветвям, будет загружен на разные конвейеры, а код, соответствующий последовательным ветвям, будет загружен только на один конвейер. При этом обработка и выполнение команд ускоряется.

Загрузка конвейера команд и параллельно функционирующих устройств осуществляется следующими устройствами:

* аппаратурой процессора

Аппаратура МП ориентирована на выделение более простых форм параллелизма из определенного количества последовательных программ. Естественный параллелизм имеет место при вычислениях целочисленных адресных выражений и обработке данных с плавающей точкой. При этом в процессе выполнения расходуется часть вычислительного ресурса ЦП на загрузку конвейера (динамическая загрузка). При использовании компиляторов, способных создавать параллельный код программ, осуществляется статическая загрузка конвейера команд.

* компилятором

В компиляторе используется более сложная технология извлечения параллелизма из последовательных программ. Таким образом, загрузка конвейера (или точнее план загрузки конвейера) составляется на этапе предшествующем началу вычислений. При этом не расходуется ресурс ЦП на динамическую загрузку процессоров (см. рисунки), а скорость обработки возрастает.

* совместной аппаратурой процессора и компилятора

Загрузка конвейера команда может быть осуществлена аппаратурой процессора и компилятора совместно, при этом частично функция загрузки реализуется статически с помощью компилятора и частично динамически с помощью аппаратуры процессора.

Дополнительным средством повышения производительности микропроцессоров является объединение процессорных элементов с несколькими внутренним конвейерами во внешний конвейер обработки данных (рисунок 45).

### Обстоятельства возникновения RISC-процессоров

При реализации второго и третьего поколения СВТ (первое поколение – на электрическом реле) было замечено, что введение в систему все более сложных команд позволяет повысить скорость обработки данных. Причина состоит в том, что сложная команда имеет один код операции и выполняет обработку сразу многих данных.

Дальнейшее развитие СВТ показало, что на повышение производительности оказывают влияние два фактора, связанных с системным программным и аппаратным обеспечением:

1. Анализ кода программ, который получается компиляторами языков высокого уровня, показал использование ими ограниченного набора простых команд форматов:

<регистр, регистр → регистр>

<регистр ↔ память>

Компиляторы не в состоянии эффективно использовать сложные команды, модифицирующие содержание групп регистров и ячеек памяти.

1. Развитие архитектуры конвейера процессоров, в частности семейство конвейерных ЭВМ Cray (Cray Research).

Если первая причина связана с системой программного обеспечения (компиляторами), то вторую причину можно отнести к аппаратному обеспечению вычислительной системе. Конвейерная вычислительная система по классификации Флинна (Flynn) относится к классу вычислительных систем типа MISD (Multiple Instruction Stream and Signal Date Stream – множественный поток команд, одиночный поток данных). В них используются отдельные наборы команд для преобразования данных в регистрах процессора. Каждая команда разбивается на небольшое количество тактов с одинаковым временем исполнения. Отдельные наборы команд для работы с памятью и преобразованием данных в регистрах процессора являются аналогами простых команд, используемых для преобразования, загрузки и разгрузки регистров центрального процессора и памяти, и обработки данных в регистрах центрального процессора. Конвейерные вычислительные системы завоевали большую популярность за счет значительных вычислительных ресурсов сложного процессора, который, по сути, состоял из ряда процессоров, используемых на стадиях обработки данных во внешнем конвейере. У такого сложного процессора имелась единая внешняя память, из которой на вход внешнего конвейера в потоки вводились данные, а с выхода в поток выводились данные в общую память. С устройства памяти команд на каждую ступень конвейера подавался сформированный поток команд. В этих потоках преимущественно используются команды для обработки данных и для работы с памятью (загрузка/разгрузка регистров).

Ступени внутри конвейера совпадают с фазами обработки команд. Каждая фаза обработки должна длиться примерно одинаковое время с целью повышения загруженности конвейера команд полезной работой. В этом случае минимизируется ожидание данных при передаче между соседними ступенями конвейера (смотри рис. 24).

Первая команда поступает на ВКИП и затем передается на ДК; при этом вторая команда поступает на ВКИП. Далее первая передается на ВАО, вторая на ДК и третья поступает на ВК и так далее (рис. 24).

В первых пяти тактах не выдается результат. Результаты, начиная с 5,6,7 тактов, будут следовать в 5 раз быстрее по сравнению со строго последовательной обработкой на одной ступени конвейера.

Таким образом, линейный конвейер осуществляет ускорение обработки команд, почти прямо пропорционально от числа ступеней.

Вывод: Можно построить конвейер команд, способный на каждом такте выдавать результат исполнения одной команды. Принципиально следующее: чем больше ступеней внутри конвейера, тем выше частота выдачи результатов и, следовательно, его производительность. Количество ступеней и производительность ограничены свойствами алгоритмического обеспечения (зависимость по данным) и характеристиками внутреннего конвейера данной вычислительной системы.

Но есть здесь и проблемы, связанные с зависимостями по данным и управлению между последовательно запускаемыми в конвейер командами, использующими результат предыдущей команды. В конвейере команд возникают пустые такты, во время которых не происходит никакой обработки, как следствие, потеря эффективности обработки конвейера. При этом компилятор, который создает исполняемый код программы, вставляет команду NOP (нет операции), которая не изменяет состояние вычислительной системы.

Пусть требуется выполнить 2 операции сложения. c=a+b, d=c+1

MOV c,a

ADD c,b

NOP

NOP

NOP

NOP

INC c

|  |
| --- |
| Рис. 46. Заполнение конвейера команд пустой командой NOP |

Зависимости по данным проявляют себя негативно при необходимости использования результата предыдущей операции. При этом необходима задержка поступления в конвейер следующей команды. Но конвейер должен функционировать для получения результата, и он заполняется необходимым количеством команд NOP.

Зависимости по управлению связаны с командами условного перехода. Для вычисления адреса перехода необходимо дождаться выполнения условий. Решение этой проблемы состоит в прогнозе ветвлений или перехода. Если центральный процессор обладает дополнительным вычислительным ресурсом, который позволяет одновременно с вычислениями в цикле произвести вычисления условий выхода из цикла, то команда ветвления или перехода будет выполняться без задержки. Можно заранее определить передачу управления на начало цикла. Это будет один и тот же адрес. Для выхода из цикла управление передается на адрес команды, который может быть заранее вычислен. При выполнении одиночной команды ветвления или сравнения, одновременно с вычислениями, предшествующими этим командам, может быть вычислено условное выражение для команд ветвления или передачи управления. Для одновременных вычислений необходимо иметь на центральном процессоре дополнительный ресурс, например в виде еще одного конвейера команд. Первый будет выполнять вычисления до команды ветвления, а второй сможет произвести операции в условиях команды перехода или ветвления.

Двумя основными архитектурами систем команд микропроцессоров, используемыми компьютерной промышленностью на современном этапе развития вычислительной техники, являются CISC и RISC.

Для RISC-процессоров (Reduced Instruction Set Computer) характерно следующее:

* сокращенный набор команд (простые команды)
* четкое отделение команды обработки от команды работы с памятью
* выполнение любой команды занимает небольшое количество машинных тактов (предпочтительно всего один машинный такт)
* использование команд фиксированной длины и фиксированного формата для упрощения операции декодирования команд на конвейере команд
* наличие достаточно большого регистрового файла (в типовых RISC-процессорах реализуется 32 и большее количество регистров по сравнению с 8…16 регистрами в CISC архитектурах), что позволяет большее время хранить в регистрах на кристалле больший объем данных и упрощает работу компилятора по распределению регистров под переменные
* преобладание трехадресного формата команд (операнд, операнд, результат), что упрощает дешифрацию и дает возможность сохранять большее время переменные в регистрах без их последующей перезагрузки

### CISC-процессоры

Процессоры с традиционным набором команд, в котором сохранились сложные команды, стали называться CISC-процессорами. В них команды имеют много разных форматов и требуют при выполнении различного числа ячеек памяти. При этом нужно определить тип команды в процессе ее дешифрации и исполнения. Это усложняет устройство управления процессором и не способствует повышению тактовой частоты до уровня RISC-процессора на той же элементной базе.

Для CISC-процессоров (CISC – Complete Instruction Set Computer) характерно следующее:

* полный набор команд (простые и сложные команды)
* сравнительно небольшое количество регистров общего назначения
* большое количество методов адресации
* преобладание двухадресного формата команд (операнд, результат)
* наличие команд обработки типа регистр-память (каждая команда производит модификацию групп регистров или ячеек памяти; результат записывается в регистр или в память)

### Области применения RISC- и CISC- процессоров

RISC-процессоры эффективны в областях, в которых могут быть использованы структурные способы уменьшения доступа к оперативной памяти. Если данные используются только для выполнения одной команды, то фактически производительность процессора определяется временем обращения к основной памяти, и использование сокращенного набора команд только ухудшает эффективность, так как требуется частая пересылка операндов между памятью и регистрами, вместо выполнения команд типа < память, память → память >.

Замечания:

1. Развитие микропроцессоров (МП) происходит при постоянном стремлении сохранения преемственности программного обеспечения и повышении производительности за счет совершенствования архитектуры и увеличения тактовой рабочей частоты.
2. Преемственность ПО и повышение производительности противоречат друг другу.
3. RISC- и CISC-процессоры в современном исполнении имеют конвейер команд. Как правило, производительность RISC-процессоров выше, чем производительность CISC–процессоров.
4. Тактовая рабочая частота CISC-процессора достигает в настоящее время 4 ГГц, у RISC-процессоров достигает 200-500 МГц. При этом в технических данных указывается значительная производительность как у тех, так и у других процессоров. Даже у RISC-процессоров, больше, чем у CISC. Для определения реальной производительности и сравнения данных двух разных процессоров необходимо произвести замер времени выполнения одного и того же программного фрагмента, например цикла. Для этой цели существуют различные наборы тестов.
5. RISC–процессоры показывают свою высокую эффективность, когда исполняемый код состоит из простых команд, выполняемых за один рабочий такт (ЦСП). CISC-процессоры – когда исполняемый код состоит из сложных команд, которые выполняются не за один, а за много рабочих тактов. RISC-процессоры транспьютеров выполняют исполняемый код состоящий из команд выполняемых за один или несколько рабочих тактов. В их системе команд имеются команды передачи данных, которые могут исполняться неопределенное время, до завершения передачи данных.

## 3.2. Компоненты информационных технологий

В качестве компонентов информационных технологий (IT) рассматриваются:

* аппаратное обеспечение;
* программное обеспечение;
* информационные носители.

### Аппаратное обеспечение

Среди микропроцессоров (МП) в настоящее время существует несколько различных типов внутренней вычислительной архитектуры:

* фон-неймановская
* гарвардская
* транспьютерная
* длинного командного слова

Существует большое количество модификаций перечисленных вычислительных архитектур.

### Фон-неймановская вычислительная архитектура

|  |
| --- |
| Рис. 47. Структура фон-неймановской вычислительной архитектуры |

В 1944-45 гг., Дж. фон Нейман, работая в США над ламповым компьютером ENIAC, сформулировал основные принципы работы компьютера, такие как:

* принцип программного управления вычислениями на процессоре. До этого управление вычислениями осуществлялось с помощью ручных переключателей
* управляющая программа и данные хранятся в оперативной памяти, процессор используется для изменения данных и модификации команд
* обмен данными и командами между памятью и процессором осуществляется через специальное устройство, которое впоследствии получило название общая шина
* и другие

Фон-неймановская ВА - архитектура общей шины (ОШ), которая состоит из шин адреса, данных и управления. На рисунке 30 изображена полностью раздельная шина адреса. Существует мультиплексирование ОШ: совмещенная ША, ШД и отдельная ШУ;

Для фон-неймановской вычислительной архитектуры характерна единая оперативная память с единым адресным пространством и один ЦП.

Принципиальные различия между раздельной и мультиплексированной общей шиной:

* пропускная способность значительно больше за счет раздельности шин;
* раздельные шины не содержат коммутаторов (переключений);
* площадь раздельной шины на кристалле больше площади мультиплексирующей шины, что существенно с точки зрения количества размещаемых на кристалле устройств.

Итак, это классическая архитектура универсальных микропроцессоров. Существуют различные модификации фон-неймановской архитектуры.

#### Модифицированная фон-неймановская вычислительная архитектура

Усовершенствованию подвергается общая шина. В её конструкцию добавляется шина ввода-вывода, к которой подключается внешнее устройство.

Внешние устройства являются медленно действующими устройствами. С целью разгрузки шины данных от передачи по ней данных на внешние устройства вводится дополнительная раздельная шина ввода-вывода. С её введением появляется возможность для передачи данных непосредственно из памяти на вычислительное устройство (ВУ) и обратно, без использования устройств центрального процессора (ЦП).

В целом производительность такой вычислительной системы возрастает. Кроме модификации общей шины, с целью повышения эффективности, модифицируются ОЗУ.

Для усовершенствования памяти применяется иерархическая память с использованием КЭШ-памяти первого и второго уровня. КЭШ-память первого уровня работает с тактовой частотой процессора, обеспечивая его данными без задержек. Объемы КЭШ-памяти первого и второго уровня подбираются специальным образом. Объем КЭШ-памяти первого уровня должен обеспечивать непрерывность обмена данными между центральным процессором и КЭШем первого уровня. А КЭШ-памяти второго уровня, должен поддерживать непрерывность обмена данными между КЭШем первого уровня и основной памятью. В КЭШ-памяти второго уровня загружаются блоки данных из ОЗУ, его объем больше КЭШа первого уровня, и, фактически, данные в КЭШ-памяти второго уровня, должны быть подготовлены заранее для использования в КЭШ первого уровня.

|  |
| --- |
| Рис. 48. Одна из модификаций фон-неймановской вычислительной архитектуры с раздельной ШВВ |

Модификация памяти не приводит к созданию изолированных ОЗУ. Адресное пространство сохраняется единым.

С точки зрения фон-неймановской вычислительной архитектуры в вычислительной системе существует единственный процессор. В этот процессор постоянно вносятся усовершенствования касающиеся системы команд (простые, сложные команды), добавляются новые устройства (такие как конвейер команд, буфер команд).

### 3.2.2 Гарвардская вычислительная архитектура

Гарвардская вычислительная архитектура (ГВА) является общим случаем фон-неймановской, поскольку также имеет общую шину. ГВА используется в специализированных процессорах, ориентированных на цифровую обработку сигналов. Практически все ЦСП построены с использованием различных видов ГВА.

Принципы построения ЦСП на основе ГВА:

* выполнение всех команд на один такт
* использование RISC-архитектуры
* раздельные ОЗУ, ША, ШД, ШК
* аппаратный умножитель в центральном процессоре (ЦП) или в процессорном ядре (ПЯ)
* введение в систему составных команд типа умножения с накоплением (с = с+а∙b)

ЦСП в своем развитии прошли 4 поколения. От поколения к поколению изменялась внутренняя архитектура, добавлялись новые устройства, возникла специализация среди ЦСП, которая позволила их использовать не только для обработки сигналов, но и для создания ВА на СБИС, многопроцессорных вычислительных систем, значительно улучшены характеристики производительности, памяти, каналов передачи данных.

Рассмотрим три возможных структуры ГВА:

1. классическая ГВА;
2. модифицированная ГВА;
3. супергарвадская ВА.

#### Классическая гарвардская ВА

Для обработки электротехнических, звуковых, видеосигналов потребовалось предложить новую внутреннюю архитектуру микропроцессоров для выполнения обработки в реальном масштабе времени. Главные усовершенствования были внесены в реализацию общей шины и ОЗУ.

|  |
| --- |
| Рис. 49. Структура гарвардской ВА |

Гарвардская вычислительная архитектура используется в цифровых сигнальных процессорах (ЦСП). Одной из основных операций является умножение с накоплением (с=с+а∙b). С целью повышения производительности ЦСП до уровня реального масштаба времени в архитектуру введены два изолированных ОЗУ. Это является особенностью такой вычислительной архитектуры: наличие двух изолированных ОЗУ и соответствующего им количества шин адреса и шин данных. Первоначально предполагалось, что ОЗУ 1 должны храниться только команды или исполняемые программные коды. В ОЗУ 2 должны содержаться только данные, которые должны использоваться в качестве операций команд записанных в ОЗУ 1. Результаты выполнения этих команд также записываются в ОЗУ 2. Увеличение скорости обработки данных происходит за счёт возможности одновременного доступа к ОЗУ 1 и ОЗУ 2 за один такт работы центрального процессора. Благодаря этому становится возможным одновременно считать из ОЗУ 1 команду во внутренний регистр центрального процессора, а из ОЗУ 2 считать один операнд (или записать в него один результат) так же за один рабочий такт центрального процессора. В последствие использование двух изолированных ОЗУ было усовершенствовано с тем, чтобы получить возможность одновременного считывания данных из каждого ОЗУ. Таким образом, в ОЗУ 1 появилась возможность для хранения команд и данных. ОЗУ 1 и ОЗУ 2 имеющие каждые своё адресное пространство может напрямую обмениваться данными с внешними устройствами с использованием ШВВ. Изолированная шина ввода-вывода позволяет подключить на ОШ ряд медленно работающих по сравнению с ЦП внешних устройств и тем самым разгрузить шины данных.

За один такт работы центрального процессора можно выбрать из памяти 2 операнда, которые записываются в регистры центрального процессора. Программное обеспечение необходимо записывать в ОЗУ 1, там же может содержаться результат выполнения команда, а ОЗУ 2 используется для хранения данных.

Центральный процессор реализован на принципах RISC-процессора. В современных ЦСП в центральном процессоре имеется буфер команд и ряд других устройств, повышающих его эффективность. Этот буфер заполняется заодно или несколько обращений к памяти за кодом команд.

Вывод: увеличение обработки скорости данных происходит за счет одновременного доступа к командам и операндам, размещенным в разных ОЗУ, за один такт работы центрального процессора. Гарвардская вычислительная архитектура используется в ЦСП, увеличение скорости обработки данных происходит так же за счет реализации принципов построения ЦСП в центральном процессоре, частью которого является дополнительные устройства, повышающие его производительность:

* аппаратный умножитель
* барабанное устройство циклического сдвига, позволяющие за один такт выполнять операцию умножения на 2n
* буфер команд, представляющий собой уровень КЭШ-памяти, в котором содержатся ближайшие следующие команды для выполнения
* два генератора адреса
* и другие

Недостаток:

* Значительная площадь на кристалле отводится под общую шины и два раздельных ОЗУ
* Площадь, на которой реализуется центральный процессор, недостаточна для размещения на ней дополнительных устройств повышающих производительность центрального процессора.

#### Модифицированная гарвардская вычислительная архитектура

В модифицированной гарвардской архитектуре ЦП начинает трансформироваться в ПЯ, имеющее устройства, повышающие его эффективность. Ряд фирм оставили на кристалле два независимых ОЗУ, имеются реализации с тремя независимые ОЗУ. Были введены дополнительные команды, обеспечивающие прямой обмен данными между ними. Эти архитектурные аспекты рассматриваются далее. Теперь стала возможна перезапись из одного ОЗУ в другое, минуя регистры центрального процессора, число которых остается достаточно небольшим. Актуальность таких команд возросла в связи с появлением третьей памяти для возможности переписывать операнды из ОЗУ 2 в ОЗУ 3 и наоборот, а также для переписывания результата из ОЗУ 1 в 2 или 3.

Концепция классической гарвардской вычислительной архитектуры (ГВА) и RISC-процессоров позволяла осуществлять обмен данными между ОЗУ с помощью двух команд, через регистры центрального процессора.

В настоящее время существуют четыре поколения ЦСП с традиционной гарвардской вычислительной архитектуры, модифицированной и супергарвардской вычислительной архитектуры.

Модифицированная гарвардская и супергарвардская вычислительная архитектура широко используются в ЦСП четвёртого поколения.

|  |
| --- |
| Рис. 50. Общая структура модифицированной гарвардской ВА |

На кристалле ЦСП располагаются все устройства за исключением внешних устройств.

#### рис7Cупергарвардская ВА

Рис. 53. Структура супергарвардской ВА

Блок ПЯ содержит в супергарвардской ВА целый ряд усовершенствований, повышающих эффективность функционирования процессора. В нем находится аппаратный умножитель, два набора регистровых файлов (а и б), конвейер команд, буфер и диспетчер последовательности команд, управляющие регистры (для ускорения выполнения команд и управления вычислительным процессом), управляющая логика. Второе новшество состояло в том, что была усовершенствована шинная архитектура с помощью включения шинного коммутатора, ориентированного на расширение коммуникационных возможностей в параллельной работе ЦСП. Впервые это было сделано фирмой Analog Devices и является стандартом в реализации СГВА (рис. 54).

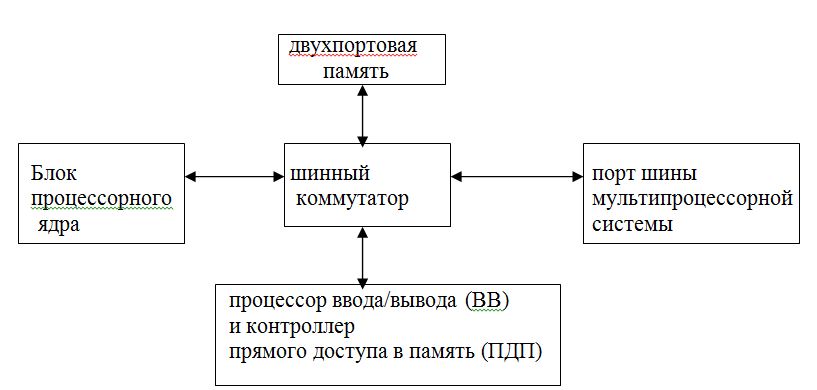


Рис. 54. Архитектура SHARC – стандарт СГВА

СГВА эффективна при построении распределенных и связанных систем и представляет собой новый стандарт для архитектуры ЦСП, которого стараются придерживаться и другие фирмы.

ЦСП ADSP 2106 имеет пропускную способность по 6 линкам, равную 240 Мбайт/c.

Интерфейсы линков имеют *последовательный байтовый протокол передачи данных*:

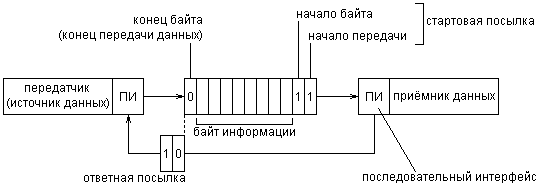
(1) *синхронный* (рис. 67) → Т414, поскольку передача данных осуществляется через регистр

Рис. 67. Синхронный последовательный байтовый протокол передачи данных

Синхронный последовательный байтовый протокол передачи данных предусматривает передачу данных с использованием регистра. ответная посылка 10 интерфейсом приемника данных выдается после записи байта информации в регистр приемника. Записанные данные должны быть использованы и переписаны из регистра приемника в память. Поступление ответной посылки в интерфейс передатчика приводит к передачи следующего байта информации вместе со стартовой посылкой и концом передачи данных. Этот процесс продолжается пока не будет передан массив данных. Синхронизация передачи информации, связанная с получением посылки, замедляет скорость передачи данных.

(2) *асинхронный* (рис. 68) → Т800, Т9000, поскольку имеется буфер памяти

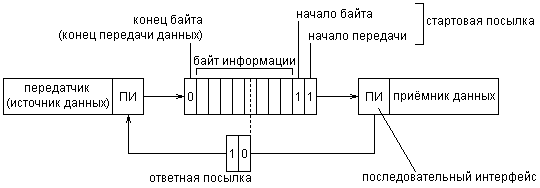
При наличии буфера памяти приёмник данных посылает ответную посылку во время приёма байта. При заполнении буфера ответная посылка не передаётся, а сама передача прерывается. За счёт совмещения времени приёма байта и передачи ответной посылки скорость обмена данными существенно возрастает (в 1,5-2,5 раза, в зависимости от длинны байтового сообщения). Ответная посылка выдается интерфейсом приемника с таким расчетом, чтобы она поступила в интерфейс передатчика к началу передачи следующего байта, таким образом исключаются задержки, связанные с ожиданием ответной посылки. С интерфейса передатчика на

Рис. 68. Асинхронный последовательный байтовый протокол передачи данных

интерфейс приемника поступает поток байтов с тремя служебными битами начала и конца сплошным потоком беспрерывно. Скорость передачи достигает пределов пропускной способности каналов передачи.

*Замечание*: последовательный интерфейс работает медленнее параллельного, поскольку

- последовательно передаёт разряды (информационные и служебные) в обоих направлениях (исключение составляет асинхронный, который является модернизацией синхронного интерфейса без существенной его переделки в аппаратной части, изменяется только задержка выдачи ответной посылки);

- для передачи одного байта информации необходимо передать восемь информационных и ещё пять служебных разрядов дополнительно (таким образом, всего 13 разрядов).

Контроль чётности и интерпретация передаваемых данных зависит от прикладного ПО, однако вполне могут возникнуть проблемы с корректностью передачи данных.

# Классификация ВС по Флинну.

На протяжении трех десятилетий развития средств вычислительной техники (СВТ)с, существовало много частных подходов систематизации ВС, однако, не существовало подхода, имевшего общий характер классификации ВС.

В 1972г (по некоторым источникам – 1966 г. ) американский специалист Флинн (Flynn) систематизировал ВС, классифицировав их по двум следующим *признакам*:

1. количество потоков команд /ПК/;
2. количество потоков данных /ПД/.

Каждой ВС ставилось в соответствие два числа (m, n), где m - количество ПК, а n - количество ПД соответственно.

Недостатки:

1. невозможно однозначно реализовать ВС по такой записи;
2. так как m и n могут быть любые целые числа, то это затрудняет построение системы.

Например, запись (3,5) неоднозначно определяет архитектуру ВС. Такой записи можно сопоставить целый набор ВС.

ПК-поток команд.

Однако по имеющейся реализации ВС можно сказать к какому классу она относится, то есть определить значения m и n. Поэтому Флинн предложил рассматривать 4 признака ВС:

1. SI (Single Instruction Stream) - один ПК;
2. MI (Multiple Instruction Stream) - много ПК;
3. SD (Single Data Stream) - один ПД;
4. MD (Multiple Data Stream) - много ПД.

Относительно этих четырех признаков, Флинну удается упростить классификацию ВС, сведя ее к 4-м относительно широким классам.

Таким образом, Флинн выделил (сформулировал) *четыре варианта ВС*:

1. SISD *=* (1, 1) - один ПК, один ПД → ОКОД;
2. SIMD *=* (1, n) - один ПК, много ПД → ОКМД;
3. MISD *=* (m, 1) - много ПК, один ПД → МКОД;
4. MIMD *=* (m, n) - много ПК, много ПД → МКМД.

Рассмотрим подробнее каждую ВС в отдельности.

* *SISD* (рис. 81, рис. 82):

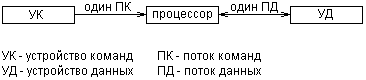
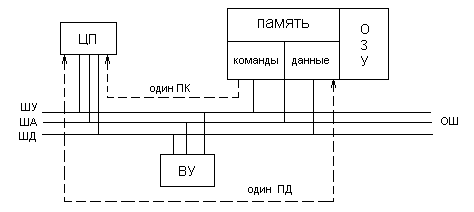


Рис. 81. Общий вид ВС типа SISD

УК – управляющее устройство, реализующееся аппаратно или управляющее устройство с памятью для команд.

УД – устройство, на котором хранятся данные и которое осуществляет операции ввода-вывода данных.



ЦП – центральный процессор

ВУ – внешние устройства

ОШ – общая шина

ПК – поток команд

ПД – поток данных

Рис. 82. Частный случай ВС типа SISD - фон-неймановская ВА

В ЦП находится АЛУ (арифметическое логическое устройство), на которое поступает ПК. Этот ПК поступает по тому же каналу, по которому поступают и данные, т. е. по ШД.

Фон-неймановская ВА имеет один ЦП и одну общую память. Потоки команд и данных поступают через ШД.

* *SIMD* (рис. 83, рис. 84, рис. 85):

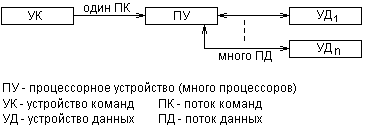


Рис. 83. Общий вид ВС типа SIMD

Одно УК управляет ПУ, состоящим из множества ПЭ, как правило, одинаковых. Все ПЭ выполняют операции или программные модули, как правило, одни и те же над разными потоками данных.

Рассмотрим частный случай ВС типа SIMD.

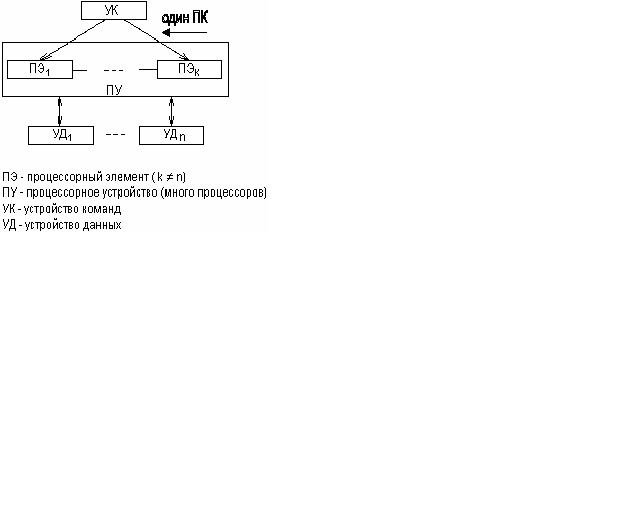


Рис. 84. Частный случай ВС типа SIMD - систолическая ВА

Систолический вычислитель основан на двух принципах функционирования:

1. конвейерная обработка данных;
2. высокая степень параллельной обработки данных.

Все ПЭ в ПУ образуют процессорную сеть.

Систолическая ВС состоит из ПЭ с локальными ОЗУ, а сами ПЭ соединяются между собой постоянными соединениями. УК через определенные интервалы времени выдает одинаковую команду на все ПЭ, например,

* начать коммуникацию;
* закончить коммуникацию;
* начать вычисления;
* закончить вычисления
* и т.д. до получения результатов.

УК достаточно просто реализуется на основе использования

1 таймера

2 выборки из памяти команд и загрузки её во все ПЭ для одновременного синхронизированного выполнения.

Один из вариантов использования гарвардской ВА:

1. Читается команда из ОЗУ1
2. При выполнении этой команды одновременно читаются данные из ОЗУ1 и ОЗУ2
3. Одновременно читается следующая команда из ОЗУ1 и записывается результат в ОЗУ2

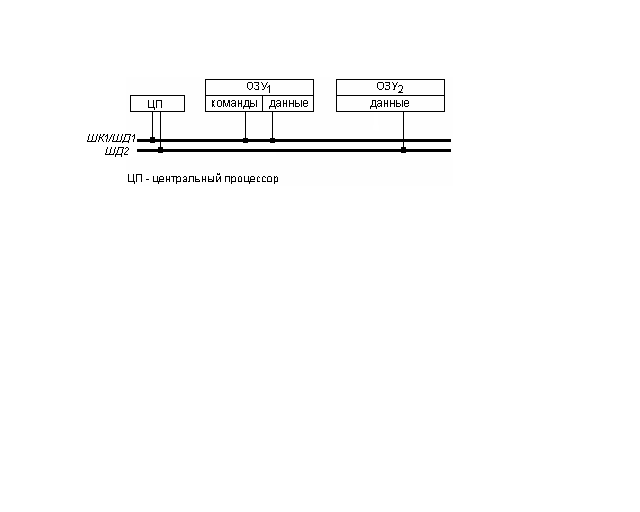


Рис. 85. Частный случай ВС типа SIMD - гарвардская ВА

В направлении ЦП возникают следующие потоки:

* один ПК;
* два ПД.

Таким образом гарвардская ВА может быть отнесена к SIMD ВА.

* *MISD* (рис. 86, рис. 87):

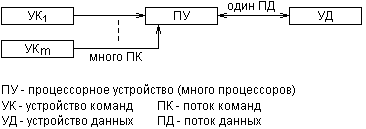
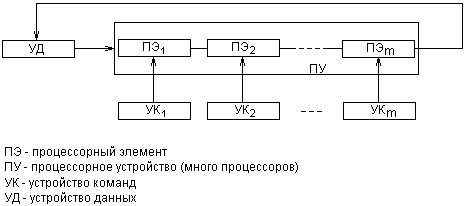


Рис. 86. Общий вид ВС типа MISD



Как правило, к этому классу относятся конвейерные ВС

Рис. 87. Конвейерный вычислитель

Принципиально, что каждая ступень конвейера может быть реализована с помощью одних и тех же ПЭ, в этом случае при полностью заполненном данными конвейере, из m ПЭ достигается производительность, которая может быть оценена величиной , где Р - производительность ПЭ. С другой стороны, каждый ПЭ может иметь различный ресурс по производительности. В этом случае максимальная производительность будет иметь более сложную функцию, зависящую от производительности каждого процессора.

Количество процессоров в ВА MISD соответствует количеству потоков команд .

Конвейерные системы основаны на разделении алгоритма на задачи последовательного выполнения. ПЭ выделяются для каждой задачи и работают параллельно. Данные вводятся и выводятся по шагам обработки. На каждом шаге обработки в каждом ПЭ может выполняться одна элементарная операция или программный модуль.

Производительность конвейера определяется скоростью работы самого медленного ПЭ, точнее программным модулем, выполняемым на этом ПЭ.

Устройства команд являются программно управляемыми. Его можно представить следующим образом. Каждый ПЭ имеет память команд, из нее команды поступают в АЛУ ПЭ и выполняются.

Устройство данных одно на весь конвейер. Данные с него поступают на ПЭ1 и возвращаются с ПЭn.

Конвейерные ВС имеют два важных преимущества:

1. регулярные связи между ПЭ конвейера;
2. простота программирования, так как программированию подлежат определенные подзадачи каждого ПЭ, а не задачи в целом.

Их существенный недостаток: алгоритмы обработки не должны существенно зависеть от данных.

Данные должны поступать на каждом шаге регулярно с тем, чтобы поддерживать его (конвейер) постоянную загруженность вычислительной работой. Если промежуточный результат одного из ПЭ требуется подать на вход конвейера, то возникает его неполная загрузка и уменьшается его производительность.

* *MIMD* (рис. 88, рис. 89):

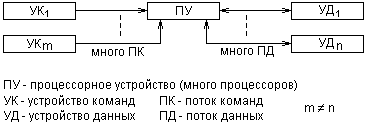


Рис. 88. Общий вид ВС типа MIMD

Независимые ПЭ выполняют различные ПМ (программный модуль). При этом они взаимодействуют друг с другом, обмениваясь данными через межпроцессорные каналы связи или общую память. ПЭ могут иметь как локальную, так и общую память, например, разделяемую. В рамках этой концепции можно создавать

* гибкие архитектуры соединений ПЭ;
* ВС с высокой производительностью;
* массово-параллельные ВС.

# 5.ВА на СБИС.

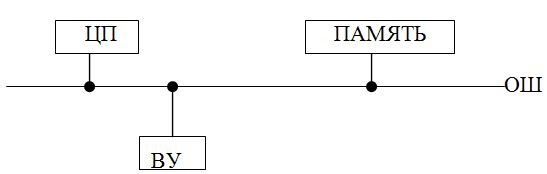
Ресурсы ВС

1. производительность
2. объем памяти
3. пропускная способность

Для однопроцессорной системы существуют ограничения сверху на производительность одного ЦП, пропускную способность ОШ и адресную память ОЗУ. Отдельные классы архитектур МП позволяют улучшить характеристики вычислительного ресурса за счет использования нескольких изолированных ОЗУ (1-3) и соответствующего количества шин адреса и данных, которые являются изолированными. Для подключения ВУ используются отдельные шины ввода/вывода на кристалле могут быть размещены несколько прцессоров. Все это характерно для Гарвардской ВА.

Три проблемы при создании современных ВС на СБИС

1. *проблема «узкого места»*:



В рассмотренной ранее фон-неймановской ВА таким «узким местом» является *ОШ*: через неё данные поступают с устройства данных в ЦП и обратно; через ту же ОШ передаются в ЦП и команды (ведь по сути это те же самые данные). Пропускная способность конкретной ОШ всегда ограничена, а любая команда при выполнении сначала считывает данные, а затем снова записывает их в память (возвращает результат). Проблему «узкого места» можно решить использованием сразу нескольких процессоров и нескольких каналов ПД, что позволит вполне адекватно повысить производительность и пропускную способность. Другой проблемой «узкого места» является ограниченная емкость запоминающего устройства.

Возможные пути решения это проблемы:

* ввести несколько каналов: канал для передачи команд (ШК) и несколько каналов для передачи данных (ШД), канал для ввода/вывода (ШВВ) для обмена данными между внешним устройством, ЦП и памятью;
* ввести дополнительное ОЗУ (все зависит от его емкости), несколько потоков команд и данных;
* использование не одного, а нескольких ПЭ (вместо одного ЦП), которые образовывали бы более сложные процессорные устройства со следующими свойствами:
* достаточное количество каналов для передачи потоков команд и потоков данных
* достаточная емкость и организация памяти
* достаточное количество ПЭ.

Необходим переход к параллельным ВС, которые позволяют решить в комплексе проблему узкого места.

2) *проблема стоимости АПК.*

Современные ВС – системы с параллельной обработкой команд и данных, причем одновременно могут обрабатываться и команды и данные.

Об эффективности ВС часто можно судить по отношению производительности ВС к ее стоимости.

Для однопроцессорной ВС с высокой производительностью характерна высокая стоимость аппаратного обеспечения. Напротив, то же соотношение производительности к стоимости для многопроцессорных ВС может быть достигнуто при меньших затратах на аппаратное обеспечение.

Эффективность ВС=производительность/стоимость

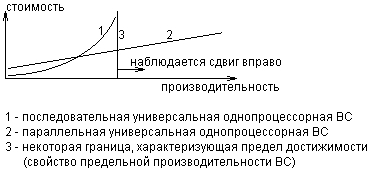


Рис. 92. Качественные зависимости для сравнения последовательной и параллельной ВС

Из графика видно, что с помощью параллельной обработки команд и данных в многопроцессорных ВС при одинаковой стоимости можно достичь значительно большей производительности как на участках до границы (3), так и после нее.

Для совершенствования разработки и технологии производства однокристальных МП требуются большие затраты, связанные с повышением производительности, использованием и созданием новых материалов и физических принципов.

3) *необходимость реализации параллельной обработки* команд и данных:

Параллельная обработка данных позволяет решить проблемы «узкого места» и стоимости. Реализация параллельной обработки команд и данных требует рассмотрение существующих концепций ВА на СБИС и соответствующих классов алгоритмов. Каждой ВА на СБИС соответствуют классы эффективных алгоритмов, выполняемых операций и используемых ПЭ-ов.

## Разработка и использование архитектурных концепций ВС на СБИС.

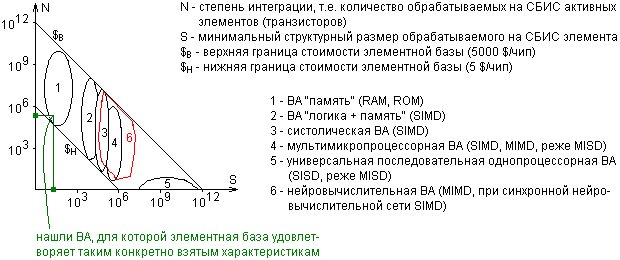


Рис. 93. Характеристики различных ВА на СБИС

ВА «память» можно отнести к SIMD или MIMD классификации по Флинну скорее по физическому принципу функционирования элементной базы и возможностям создаваемых на ее базе ВА.

Каждая *область* на графике рис. 50 - некая ВА, причём границы областей чисто качественные. Это означает, что не все численные значения оказываются такими уж точными, зато основные соотношения показаны правильно. Возьмём одну некоторую *точку*, например, внутри области 1. Она имеет две координаты, соответствующие количеству обрабатываемых на СБИС элементов и техническим нормам проектирования в виде относительного изменения минимального структурного размера элемента. Это означает, что на практике существует такая СБИС, которая может быть использована для реализации данной ВА. Причём области на графике пересекаются, а это означает, что данную ВА могут создавать сразу несколько различных элементных баз.

ВА на СБИС состоят из наборов СБИС, функционально выполняющих вычислительные, логические и другие операции, в том числе по доступу к памяти, и конструктивно представляют собой комплексы, состоящие из структур памяти, структур с перестраиваемой логикой и процессорных элементов.

## Краткая характеристика ВА на СБИС.

При дальнейшем рассмотрении каждой ВА на СБИС будем обращать внимание на три следующие её *характеристики* (показателя):

1. элементная база, используемая для непосредственной физической реализации;
2. реализуемые данной ВА операции;
3. реализуемый данной ВА класс эффективных алгоритмов.
4. Для каждой из 6-ти ВА имеются наиболее эффективные для ее реализации элементные базы. С другой стороны, в случае пересекающихся областей ВА можно сказать, что одна и та же элементная база может быть использована с различной эффективностью для построения пересекающихся ВА. Например, точка, принадлежащая областям 3, 4, 6, может быть ЦСП, который применен к реализации этих ВА. Например, транспьютер эффективен для 4 ВА и с меньшей эффективностью для 3 и 6. Для 3 наиболее эффективны систолические процессоры, а для 6- нейрочипы.
5. ВА(1) – реализует только операции доступа к памяти

ВА (2)- используются операции, которые присутствуют в алгоритме, а алгоритм реализуется с помощью информации, записанной в память об операциях.

ВА (3) – экстремально высокая скорость выполнения арифметических операций. Выполняются операции алгоритма также, как ВА (2). Арифметические, логические и другие операции, а также операции доступа в память. Обработка данных и коммуникация. Выполняется также:

ВА (4), которая реализует широкий набор операций.

ВА (5)- как ВА (4).

ВА(6) реализует основные элементарные операции и операции соединения между отдельными слоями нейронов по передаче данных.

1. ВА (1) реализует алгоритм, связанный с доступом к данным.

ВА(2)- классы жестких непрограммируемых алгоритмов. «Жесткие» алгоритмы используются с целью подчеркнуть очень узкий класс реализуемых алгоритмов с помощбю аппаратного обеспечения (термин).

ВА(3)- класс «жестких» систолических алгоритмов, обладающих свойствами синхронности, распределенности и регулярности для процессоров и их коммуникационных структур. Существуют их каталоги.

ВА (4)- вычислительные алгоритмы:

* с алгебраическим параллелизмом,
* геометрическим параллелизмом
* коллективного решения.

ВА (5) – наиболее широкий класс алгоритмов, обладает гибкостью, универсальностью.

ВА (6)- эффективно реализуемый класс нейроалгоритмов.

Для 3-ей, 4-ой и 6-ой ВА исходное АО последовательного алгоритма должно быть преобразовано к соответствующему систолическому распределенному потоку и нейро-вычислительному виду. 2-ая ВА создает аппаратное обеспечение, способное реализовать конкретный алгоритм обработки данных

* *ВА «память»* - условное название структур интегральной памяти (RAM, ROM):

1. БИС и СБИС ОЗУ, с помощью которых можно создать (реализовать) устройства на основе

* кэш-памяти;
* регистровой памяти (просто регистров);
* оперативной памяти:
* внутренней (на кристалле процессора);
* внешней (вне кристалла процессора соответственно).
  1. операции доступа к памяти (запись, перезапись, чтение);

арифметические операции («+», «-», «\*», «/») не производятся.

Для реализации операции записи сигнала нужно произвести операцию выбора адреса, а затем записать данные по тому же адресу. Операция перезаписи - по выбранному адресу считываются данные, а затем по тому же адресу записываются новые данные. Операция чтение - производится после определения (задания) адреса.

Обычно при операции доступа к памяти можно одновременно производить несколько операций чтения. При записи и перезаписи, как правило, другие операции доступа к памяти запрещаются. Арифметические операции не производятся.

3) обработка массивов данных (используется т.н. массивная память);

выделение (поиск, выборка) определённых данных (информации);

изменение определённых данных.

К 1): существуют 2 вида интегральной памяти:

— статическая (SRAM). Реализуется с помощью триггеров. Для одного бита информации требуется 4-6 транзисторов. Является более быстродействующей по сравнению с динамической памятью.

* динамическая (DRAM). Реализуется с помощью емкостной связи, представляющей собой конденсаторные ячейки. Они, как и транзисторы, имеют 2 устойчивых состояния – заряжен и незаряжен, которые используются для хранения бита информации. Для управления зарядом и разрядом конденсатора необходим лишь электронный ключ, поэтому объем оборудования для хранения одного бита информации примерно раза в 4 меньше, чем при реализации статической памяти. Для предотвращения потери информации ввиду постоянного разряда конденсаторов током утечки необходимо периодически проводить считывание состояние заряда конденсатора с последующим восстановлением этого заряда путем записи. Такой процесс постоянного считывания и перезаписи заряда называется регенерацией памяти и требует дополнительного времени в цикле обращения к памяти.

В настоящее время более медленная динамическая память служит основой нижнего уровня иерархии памяти ВС.

***Замечание:*** Существует еще одна разновидность памяти - видеопамять. В ней наблюдается 2 тенденции:

1. увеличение быстродействия памяти, увеличение объемов памяти
2. реализация видеопамяти совмещается на одном чипе с количеством обрабатывающих устройств примерно 10000.

ВА «память» *применяется* в информационных и информационно-справочных системах, базах данных для обработки данных в системах с локальной и общей памятью, а также в интерфейсных системах (ВУ подключены, например, к ПЭВМ на ОШ), где может быть большая буферная память и специальная структура памяти (например, магазинная ленточная).

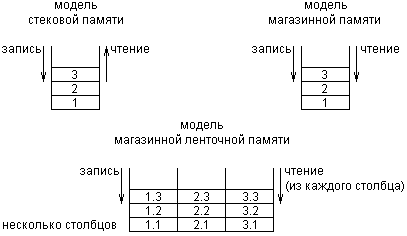


Рис. 94. Некоторые модели памяти

Отличие модели стековой памяти от магазинной состоит в том, что в стековой памяти запись производится в прямом порядке, а чтение в обратном; в магазинной и запись, и чтение – в прямом порядке. Таким образом, элемент, записанный в стековую память последним, будет извлечен из нее первым, а в магазинную память элемент, записанный первым, извлекается также первым.

Модель магазинной ленточной памяти представляет собой объединение нескольких моделей магазинной памяти, в которых записываются или считываются несколько потоков данных.

Модель магазинной ленточной памяти эффективна при подключении систолических процессорных полей к интерфейсным системам.

Для увеличения до 10 раз производительности ВС используют различные *структуры памяти*:

д) *расслоение памяти* - это структурный способ уменьшения времени доступа к памяти (рис. 57)

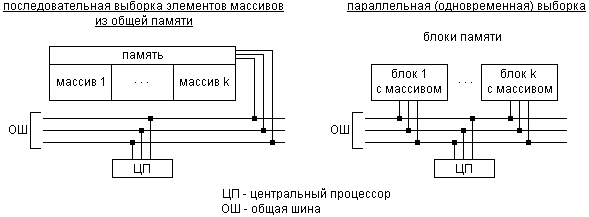


Рис. 100. Расслоение памяти

* *систолическая ВА* - - однородное процессорное поле синхронно работающих процессоров, связанных между собой регулярными коммуникационными соединениями (SIMD):

Однородное процессорное поле состоит из ПЭ одного семейства или одного типа, которые можно соединять между собой напрямую посредством имеющихся на кристалле интерфейсов. Таким образом соединения ПЭ между собой упрощаются: становятся более дешёвыми, в частности из-за того, что не требуется использование больших интегральных схем БИС. Регулярное коммутационное соединение позволяет напрямую соединить два ПЭ, причём эти соединения остаются постоянными в данной аппаратной реализации.

Рассмотрим систолическую ВА по схеме:

1. элементная база

* БИС и СБИС систолических микропроцессоров /СМП/;
* БИС и СБИС ЦСП;
* заказные СБИС, реализующие систолические алгоритмы

На практике встречаются типы СМП:

А) Непрограммируемые СМП: на кристалле имеется набор из трёх регистров:

- регистр «а» – операнд

- регистр «в» – операнд

- регистр «с» - результат

На кристалле имеется умножитель и сумматор. Входные данные записываются в регистры «а», «в» и «с». Сначала производится операция умножения данных, поступающая в регистры «а» и «в», а затем содержимое складывается с содержимым «с» и образуется результат:

С = А \* В + С

Данный вид СМП является непрограммируемым и реализует только операцию умножения с накоплением

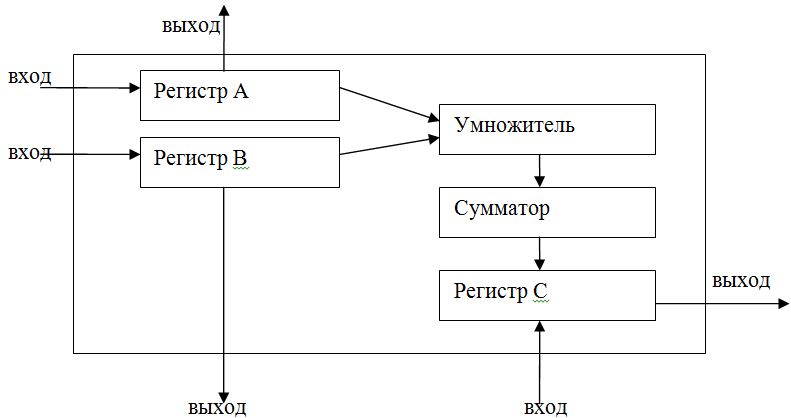


Рис.101. Схема непрограммируемого СМП

Данное СМП имеет 3 входа и 3 выхода, которые могут быть использованы для объединения СМП в процессорные полевые структуры

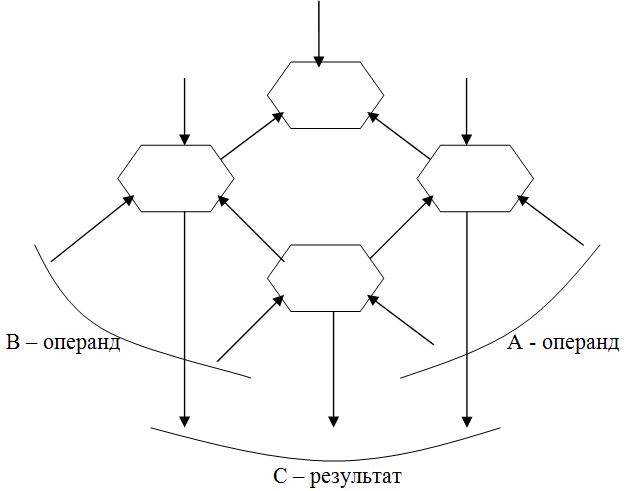


Рис. 102. Соединение СМП в процессорном поле

Из рисунка видно, что с помощью входов и выходов можно создавать регулярные коммуникационные соединения между непрограммируемыми СМП , которые способны реализовывать определённый алгоритм обработки данных. В частности данная структура предназначена для реализации умножения двух ленточных матриц «А» и «В», каждая из которых содержит по две ненулевых диагонали

Б) Программируемые СМП.

Могут содержать в системе команд большое количество арифметических, логических операций ввода – вывода и других операций (до 256). Они имеют память, состоящую из ПЗУ и ОЗУ. В них могут быть записан как программный код по обработки данных, так и сами данные. Так же как в непрограммируемых СМП программируемые имеют на кристалле реализованные интерфейсы, поддерживающие параллельный байтовый обмен данными между соединёнными СМП. Таким образом данный вид СМП позволяет реализовывать некоторые запрограммированные алгоритмы обработки, записанные на кристальную память. Объединение СМП в процессорное поле позволяет реализовывать более сложные алгоритмы обработки данных.

На практике встречаются более сложные типы систолических процессоров: имеющие на кристалле значительно количество элементов.

- N64000 – фирма Inova; на кристалле 80 ПЭ, из них 64 основных и 16 резервных. Каждый ПЭ имеет по 9 16 – ти разрядных умножителей и 32 – х разрядный сумматор, 4 кб ОЗУ для 8 – ми или 16 – ти разрядных весовых коэффициентов, 32 регистра. Этот чип представляет собой внутреннюю организацию процессоров с типичной SIMD архитектурой (реализуют всего 1 поток команд, одновременно управляющих работой всех процессоров).

* МА 16 – чип фирмы Siemens, реализующая операции над матрицами 4 x 4, на котором имеются умножители и сумматор – 16 – ти разрядные. Для чипа разрабатывается внешнее ОЗУ для хранения локальных данных.

Оба эти СМП могут быть использованы для реализации нейровычислительных архитектур.

Среди ЦСП:

* TMS 320C4\* (Texas Instruments);

В это семейство входят ЦСП: TMS 320C40, TMS320C44, TMS320LC40( низкое энергопотребление)

C44 представляет собой более дешёвый вариант по сравнению с C40, например имеющий только 4, а не 6 внешних коммутационных каналов линком, но туже производительность. LC40 имеет большую производительность ( на 30% лучше чем C40 и C44)

* ADSP 2106\*, ADSP 2116\* (Analog Devices);

ЦСП фирм Texas Instruments и Analog Devices соединяются посредством Линков в мультипроцессорной системе, образующие ВА.

* DSP 96001 (Motorola).

Посредством двух идентичных портов расширения памяти, которые могут быть преобразованы в хост интерфейс, обеспечивается интеграция процессора в мультипроцессорную систему.

Эти ЦСП могут быть эффективно использованы для построения систолической ВА, благодаря имеющимся внешним интерфейсам – внешним линкам связи. При их использовании СВА реализуется на одной или нескольких платах.

**Замечание:** Для эмуляции СВА можно использовать семейство транспьютеров Inmos Ltd, однако их применение не столь эффективно в виду того, что внешние интерфейсы Линков подделживают последовательную байтовую передачу данных.

Заказные СБИС реализующие систолические алгоритмы.

Технология производства СБИС позволяет на одном кристалле разместить до 1000 32-х разрядных процессоров, таким образом создаются возможности для создания узкоспециализированных систолических ВА, способных реализовывать один или нескольких классов алгоритмов вычисления или обработки данных.

1. Реализованный набор операций.

В систолической ВА реализуется расширенный набор операций:

* операции доступа к памяти (запись, перезапись, чтение);
* арифметические операции (все элементарные операции «+», «-», «\*», «/»);
* логические операции («И», «ИЛИ», «НЕ», операция сравнения);
* все другие операции, определяемые системой команд СМП или ЦСП.

Операции связанные с вычислением и обработкой данных, а так же операции с предачей данных образуют вычислительные и коммуникационные модули, которые выполняются в процессорной сети синхронно. Например: сначала выполняются вычислительные модули, затем только коммуникационные, снова вычислительные и т.д.

1. эффективные алгоритмы

Рассмотрим класс эффективных систолических алгоритмов относительно 8 – ми предметных областей:

* линейная алгебра → умножение матрицы на вектор, перемножение матриц, в том числе ленточных, решение треугольной системы линейных алгебраических уравнений, LU – преобразование; нахождение обратной матрицы, собственных значений и т. д.
* обработка сигналов → быстрое преобразование Фурье, одномерные и двумерные свертки, фильтрация сигналов и др.
* вычисления в конце многочленов и компьютерная алгебра → умножение и деление многочленов, отыскание наибольшего общего делителя, отыскание наименьшего общего кратного, др. алгоритмы.

Алгоритмы компьютерной алгебры связаны с преобразованием последовательностей, формул, приведением общих членов.

* алгоритмы линейного программирования → алгоритмы методов оптимизации, в том числе симплекс – метод, линейное программирование, используются многие алгоритмы линейной алгебры, например, определение системы линейно – зависимых векторов, ранга матрицы, нахождение выпуклых многоугольников, решение систем линейных алгебраических уравнений, нахождение обратной матрицы и т. д.
* задачи идентификации → вычисление редакционного расстояния, выявление вхождений, нахождение максимального общего подслова, выявление повторений в слове, распознавание контекстно-свободных языков (КС - языки), регулярное вхождение и др.
* задачи на графах → отыскание транзитивного замыкания, выявление компонент связности и несвязности, отыскания минимального остова дерева (каркас), построение матрицы кратчайших расстояний и др.
* вычислительная геометрия → построение выпуклой оболочки точек на плоскости, триангуляции, построение пересечений многоугольников, построение ближайшей точки к плоскости и др.
* сортировка и поиск → последовательная и параллельная сортировка, слияние, например, образование сочетаний из символов и слов, образование составных слов, словарный и табличный поиск, организация стеков, очередей и др.

Замечания:

* 1. выше перечислены не все классы и не все алгоритмы;
  2. при проектировании алгоритмов стремятся осуществить декомпозицию задачи на ряд более простых подзадач. При этом удается выявить простые алгоритмы, для которых имеются либо сведения в литературе, либо реализация в каталоге систолических алгоритмов;
  3. при проектировании вычислительных систем с использованием ряда систолических вычислителей в качестве сопроцессоров стремятся определить базовый набор алгоритмов, который позволит повысить эффективность, значительно ускорить вычисления за счет аппаратной реализации алгоритмов на 1 – 3 порядка, т. е. в 10 – 1000 раз;
  4. наличие каталога реализованных алгоритмов позволяет не заниматься их разработкой;
  5. кроме базового набора эффективных алгоритмов имеются прикладные систолические алгоритмы, реализующие довольно сложные прикладные задачи для различных технических систем, например, в технике, системах управления и др.

Пример: преобразование координат точки из одной СК в другую;

позиционирование захвата робота – манипулятора;

предварительная подготовка матриц преобразования координат.

* 1. систолический вычислитель заменяет соответствующее ПО и реализует его на 1 – 3 порядка быстрее.

***СВА имеет:***

* алгоритмическое обеспечение (АлО)
* информационное обеспечение (ИО)
* аппаратное обеспечение (АпО)

АлО СВА должно быть преобразовано к специальному виду, в котором находят отражение принципы систолической обработки данных.

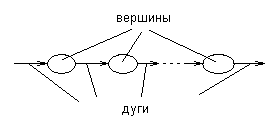
ИО СВА предполагает разработку многих потоков входных, выходных и промежуточных данных. СВА относят к классу SIMD ВА.

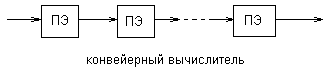
АпО СВА предполагает разработку процессорной сети и соответствующего интерфейсного устройства, в котором имеется устройство управления, выдающее на все процессоры СВА один поток команд, и устройство данных, реализующее много потоков данных на процессорную сеть СВА.

Встречаются разные определения СВА. Часто СВА называют систолическим вычислителем.

***Определение 1:*** *систолический вычислитель (СВ)* – конвейерный вычислитель с регулярным графом, подразумевающий неизменную геометрию коммуникационных соединений. При этом на каждом из ПЭ каждый раз выполняются одни и те же вычислительные операции (или вычислительные модули с разными функциями) над поступающими данными.

Регулярный граф:





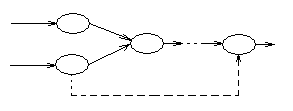
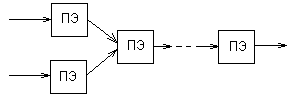
 

Рис. 103. Пример соответствия графа структуре конвейерного вычислителя.

Конвейерный вычислитель изменяет сложность в зависимости от сложности регулярного графа.

Конвейерный вычислитель представляет собой сеть конвейерных элементов, для которых характерны постоянные соединения. Можно с помощью прямого отображения на конвейерный вычислитель отобразить соответствующую структуру регулярного графа.

Каждый ПЭ реализует свою функцию, которая характеризуется сложностью вычислительного процесса и временем вычисления.

При одинаковой сложности вычислений время работы для каждого ПЭ одинаковое и регулярный граф систолического вычислителя обеспечивает равномерную загрузку ПЭ в сети. При этом потери на ожидание данных минимизируются, эффективность наивысшая для данного времени работы ПЭ.

Важным свойством СВ является регулярность или постоянство внешних соединений ПЭ, причем эти соединения между различными ПЭ невозможно разъединить, что существенно упрощает и удешевляет вычислители, т. к. не требует дополнительных коммутаторов.

***Определение 2:*** *СВ* – синхронная распределительная вычислительная структура с параллельной и поэтапной конвейеризацией вычислений, с простым и локальным управлением процессов вычислений и коммуникаций, простой и регулярной локальной геометрией структурных соединений между простыми процессорами одного семейства, имеющими малые постоянные объемы памяти для программных модулей и данных алгоритмов.

*Замечания по определениям 1 и 2:*

* определение 1 связано с представлением АО в виде вычислительного графа, свойства которого переносятся на АО;
* определение 2 связано со свойствами АО, которые присущи СВ:
* распределенность и синхронность вычислений, т. е. в один и тот же момент времени в процессорной сети выполняется одна и та же операция, либо одинаковый модуль (либо вычислительная операция, либо операция по обмену файлами);
* использование достаточно простых ПЭ;
* организация только постоянных соединений между ПЭ, обладающих незначительной памятью и способностью выполнять лишь короткие программные модули или циклы.

Систолические модули, как правило, реализуют очень ограниченный набор операций, обладают узкой специализацией, заменяют частое или интенсивное использование программных модулей, реализуются аппаратно, позволяют ЦП обращаться к ним как к сопроцессору, высвободить память от соответствующих программ и данных, ускоряют выполнение на 1 – 3 порядка, имеют высокое соотношение производительность/стоимость.

## Принцип систолической обработки данных

Рассмотрим принципиальные моменты систолической обработки данных.

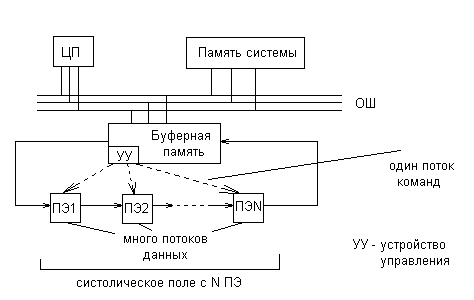


Рис. 104. Принцип систолической обработки данных.

Учитывая то, что все ПЭ одинаковы, суммарная производительность сети будет прямо пропорциональна числу ПЭ и составляет такую величину:



Заметим, что ПЭ могут быть размещены на процессорной плате с помощью чипов ПЭ или в заказной СБИС. Процессорная сеть определяется термином Systolic Array – систолический массив ПЭ, систолический вычислитель, систолическая структура (систолический алгоритм – «жесткий» алгоритм, реализованный аппаратно в процессорном поле).

Термин систолической обработки данных заимствован из медицины. При этом буферная память отождествляется с сердцем, сеть ПЭ – с кругом кровообращения, а кровоток – это потоки данных. Систолическая ВА работает ритмично, как сердце.

Суммарная производительность достигается при следующих условиях:

1. все ПЭ одинаковы или одного семейства – свойство однородности вычислительной сети. Позволяет напрямую соединить ПЭ, отказавшись от всех промежуточных интерфейсных элементов и от электронного коммутатора. При этом скорость передачи данных между ПЭ максимально возможная, АпО упрощается, стоимость реализации понижается.
2. внешние коммуникационные соединения между ПЭ осуществляются посредством одинаковых параллельных интерфейсов. Расстояния между ПЭ должны быть приблизительно одинаковыми (так называемый принцип близкодействия). Он позволяет получать максимально возможную плотность упаковки на кристалле СБИС или на процессорной плате. Время передачи внешних данных между ПЭ соответствует времени доступа к внутренним данным внутри ОЗУ.

При этом . Параллельный интерфейс обладает большей пропускной способностью по сравнению с последовательным.

1. внешние коммуникационные соединения постоянны – свойство регулярности сетевой структуры (не требуются электронные коммутаторы, вносящие задержку в передачу данных).
2. сеть ПЭ является синхронной. Свойство синхронности позволяет в один и тот же момент времени выполнять одну и ту же операцию на всех ПЭ. Однако необходима различная синхронизация в узком и широком смысле.

*Синхронизация в узком смысле* – выполнение одинаковых операций в процессорном поле в каждый момент.

*Синхронизация в широком смысле* – предполагает выполнение одинаковых вычислительных или программных модулей для любого момента времени в каждом ПЭ процессорного поля.

Можно выделить следующие фазы систолического вычислительного процесса:

* *фаза коммуникаций (К)* → осуществляется внешняя передача данных следующим образом: в отведенный интервал времени в процессорной сети происходит одновременный обмен данными между всеми ПЭ.
* *фаза вычислений (В)* → в отведенный интервал времени на всех ПЭ одновременно осуществляются как вычисления, так и обработка информации.
* *фаза управления (У)* → в отведенный очень короткий интервал времени осуществляется управление работой систолического поля. Выполняются операции, связанные с функционированием вычислительной сети: старт, стоп процессорного поля; переключение фаз К и В; останов процессорного поля в случае ошибки.

Самая короткая – У, самая длительная – В. Во время работы систолического поля длительность фаз для конкретного алгоритма постоянна. Синхронная распределенность систолической обработки данных может быть проиллюстрирована на графике:

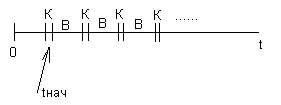


Рис. 105. Фазы систолического вычислительного процесса

В связи с этим УУ может быть реализовано достаточно просто на основе использования программируемого таймера, который для каждой из фаз задает интервал времени. Длительность всех фаз К, В и У должна быть постоянна. При этом для различных ПЭ и различных систолических полей, а также алгоритмов подбираются свои значения.

Подбор этих значений следующий:

* длительность фазы К – наиболее долгая операция по обмену данными между ПЭ сети;
* длительность фазы В – соответствует наиболее продолжительному выполнению по времени вычислительного модуля в ПЭ-х сети.

Соотношение фаз: .

***Замечание:*** в систолической ВА имеется только один поток команд, так как управляющие сигналы поступают сразу на все процессорные сети от УУ. В той же процессорной сети имеется несколько или много потоков данных. Таким образом, СВА относится по Флинну к классу SIMD.

Выполнение операций бывает:

* синхронное;
* асинхронное.

1) *синхронное выполнение* → в каждый момент времени всеми процессорами сети выполняется либо фаза К, либо фаза В (одинаковые операции, рис. 52). При этом необходимым условием является наличие централизованного вычислительного модуля или аппаратного устройства управления /УУ/, осуществляющего синхронизацию всех операций или фаз вычислительного процесса в сети.

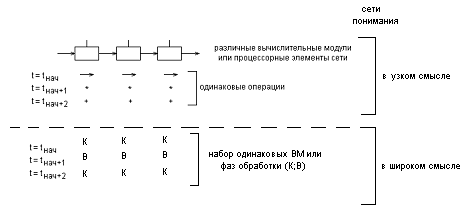


Рис. 106. Синхронное выполнение операций и модулей

(III – ВА на СБИС)

2) *асинхронное выполнение* → в каждый момент времени процессорами сети выполняются различные операции (рис. 64). При этом обычно отсутствует централизованный вычислительный модуль или УУ, в функции которого входит синхронизация по выполнению операций или модулей в процессорной сети.

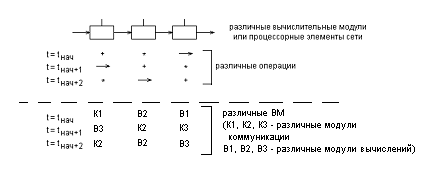


Рис. 107. Асинхронное выполнение операций и модулей

(IV – ВА на СБИС)

Рис. 62 иллюстрирует синхронную систолическую обработку данных, относящуюся к классу SIMD по Флинну.

Рис. 63 иллюстрирует мультимикропроцессорную обработку данных, относящихся к классу MIMD по Флинну.

### Применение систолических полей

Существует три наиболее развитых *направления применения систолических полей*:

1. инструментальная разработка аппаратного и программного обеспечения СВ;
2. реализация аппаратного вычислителя-ускорителя на встроенных в ПЭВМ платах: СВ рассматривается в качестве сопроцессора ВС, который аппаратно заменяет выполнение соответствующего ПО на ЦП. При этом время вычислений сокращается на 1..3 порядка;
3. СМП или спецпроцессоры, встроенные в технические системы, используются для цифровой обработки в реальном масштабе времени.

Очень часто постановка задачи следующая: разработать минимальный *базовый набор СВ*, который обеспечит значительное повышение реальной производительности. В качестве базового набора принимаются (на основе каталогов) известные систолические поля, реализующие прикладные задачи. Как правило, в каталогах содержатся базовые алгоритмы.

Для подключения систолических полей к вычислительной или технической системе необходима *интерфейсная система*, которая бывает двух типов:

1. у*ниверсальная интерфейсная система* (рис. 65):

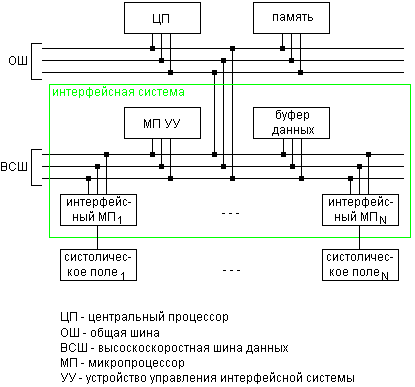


Рис. 108. Универсальная интерфейсная система

* *МП УУ* → обеспечивает корректную передачу данных между ПЭВМ и буфером номинальных данных интерфейсной системы;
* *буфер данных* → хранит исходные данные, результаты вычислений в систолических полях;
* *интерфейсные МП* → обеспечивают управление вычислениями в систолических полях (фазы К и В), поддерживают корректную передачу данных между буфером и систолическим полем (ввод/вывод данных, контроль возникающих ошибок);
* *ВСШ* → поддерживает высокую пропускную способность обмена данными между буфером и несколькими систолическими полями, поскольку входные данные на систолическое поле и результаты вычислений должны передаваться одновременно.

Интерфейсная система рассматривается главной ВС как внешнее устройство. Главная ВС и интерфейсная система могут функционировать одновременно.

*Замечание*: универсальные интерфейсные системы вполне могут быть использованы для пунктов 1, 2 применения, а третье применение требует создания более благоприятных условий для реализации потенциала систолической обработки.

(2) *специализированная интерфейсная система*

Такой интерфейс поддерживает работу одного или нескольких систолических полей и обеспечивает минимальные потери времени, не связанные с вычислительным процессом. Для организации правильного пространственно-временного потока данных на систолическое поле нужно иметь:

* специальным образом организованную память;
* УУ такой памятью, выполняющее как операции ввода/вывода данных в/из систолического поля, так и синхронизацию работы процессоров сети по фазам К и В (в качестве УУ можно использовать универсальный программируемый МП).

Рассмотрим использование ленточной магазинной памяти в качестве составного устройства специализированного интерфейса системы.

***Пример использования*** специальной ленточной магазинной памяти для выполнения матричной операции А\*В=С, где А – матрица, В,С – вектора в систолическом поле.

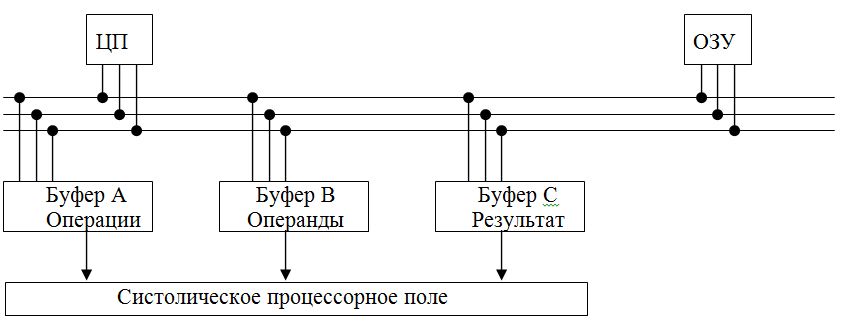


Рис. 109. Организация вычислительной среды

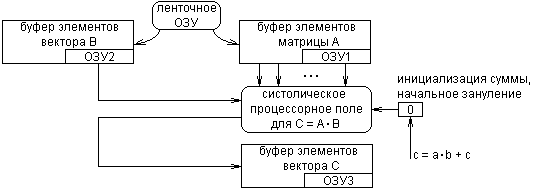
Для эффективного обмена информацией в систолической структуре ПЭ может быть использована *буферная ленточная память магазинного типа* (рис. 42), состоящая из входного буфера с исходными данными и выходного буфера с полученными результатами их обработки (рис. 65).

Рис. 110. Использование ленточной памяти в качестве буферов потоков данных

Рассматриваемая операция используется для реализации цифровых фильтров с помощью систолического процессорного поле (рис. 55) при следующей матричной постановке задачи:

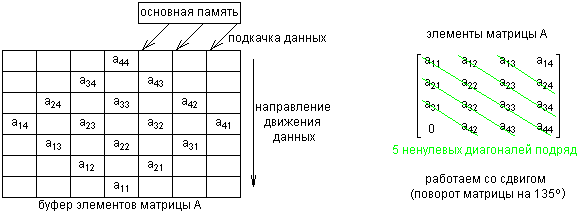


*А* - ленточная пяти-диагональная матрица (имеет 5 ненулевых диагоналей подряд);

*В* - вектор исходных данных;

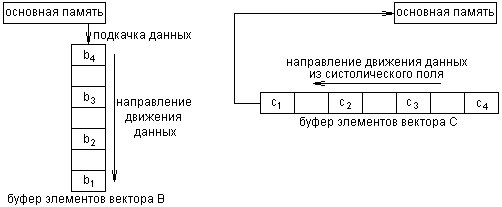
*С* - вектор полученных результатов.

Рассмотрим буфер элементов матрицы А:



Если емкость ленточного буфера памяти достаточна, то загружаем всю матрицу в буферную память. Если же недостаточно, то в процессе обработки данных в СП интерфейс МП осуществляет подкачку данных в буферной памяти. В буферной памяти каждый такт фактически происходит сдвиг данных на одну ячейку в направлении СП во время фазы К.

Рассмотрим организацию ленточной магазинной памяти для вектора В исходных данных и вектора С - результатов.



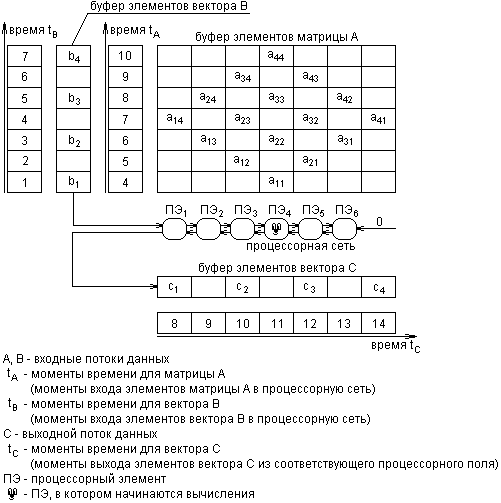
Управляющий интерфейс МП переписывает данные из основной памяти в буфере элементов.

Во время фазы *коммуникации* (внешней передачи информации) данные поступают из ленточной памяти в систолическое поле (рис. 55) или одновременно выводятся из него в ленточную память.

Во время фазы *вычислений* в ленточной памяти возможно следующее:

* содержимое ячеек памяти не меняется;
* происходит обмен с ОЗУ более высокого уровня

(при этом производится дополнение данными или считывание полученных результатов).

Изобразим диаграмму, на которой вместе покажем входной, выходной потоки данных, процессорную сеть и синхронизацию процессов коммуникации и вычислений (рис. 56).

Эта схема верна для случая а41=0. В этом случае матрица А является 6ти диагональной ленточной матрицей, ее элементы поступают в 6 ПЭ.

Рис. 111. Систолическое процессорное поле

Синхронизация входных и выходных потоков по времени показывается относительно их моментов времени ввода и вывода в/из СП.

Элементы входных потоков вводятся один раз в процессорную сеть. При этом элементы вектора В используется в процессорной сети многократно. Выходной поток результатов движется в противоположном направлении по отношению к потокам элементов вектора В.

Обобщенная вычислительная модель СП описывается тройкой (G, F, T), специфицирующей G- геометрию процессорной сети (глобальная модель), F-процессорную функцию для G (локальная модель), Т- время выполнения для F (модель управления).

Таким образом, процессорное поле создает глобальную модель, локальная модель создается процессорной функцией для каждого процессорного элемента.

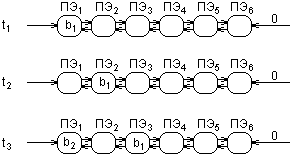


Модель системы управления определяется заданием интервалов времени для выполнения операций фаз коммуникации tk и вычислений tb.

Например: tk=1 мкс; tb= 20 мкс;

Рассмотрим теперь организацию вычислительного процесса , для чего изучим *алгоритм работы процессорного поля*. Согласно рис. 66, элементы матрицы *А* поступают в процессорную

сеть, начиная с 4-го момента времени, а элементы вектора *B* - в 1-ый, 3-ий, 5-ый и 7-ой моменты.



Моменты времени , , :

вычислительные операции ещё не осуществляются, только коммутация (элементы ,  движутся по процессорной сети).

В дальнейшем максимально в каждый момент времени в вычислительном процессе оказывается задействованы лишь половина всех имеющихся ПЭ сети (через один): в нечётные моменты работают нечётные ПЭ, в чётные - соответственно чётные ПЭ.

после рис38_4

Момент времени :

,

поскольку перед вычислениями процессорная сеть инициализируется, а все регистры обнулены.

после рис38_5

Момент времени :



после рис38_4

Момент времени :



после рис38_5

Момент времени :



Момент времени :

 вычислен и выведен за пределы процессорной сети, после чего записан в буферную память, а в ,  и  проводятся вычисления.

Остальные искомые элементы (, , ) будут выведены на 10-ом, 12-ом и 14-ом моментах времени соответственно (см. рис. 66).

*Замечания*:

1. *потенциальная производительность* процессорной сети в нашем случае вообще-то должна составлять шестикратную производительность одного ПЭ (), однако *реальная производительность* будет в два раза меньше вследствие использования лишь примерно половины имеющихся ПЭ () для вычислений на каждом шаге (в каждый момент времени).

Причина - невысокая загруженность ПЭ в вычислительной работе.

1. *ускорение вычисления* элементов вектора *C*, достигаемое в процессорной сети, составляет

 где

 - количество проведённых операций сложения и умножения (соответственно 16 +16 = 32);

 - количество моментов времени работы сети, потребовавшееся для вычисления вектора *C* (в каждый момент времени производилось вычисление ).

Стоит отметить, что полученное выше ускорение не учитывает выполнение операций присваивания (операций доступа к памяти).

При последовательном выполнении на фон-неймановском процессоре количество операций ввода элементов матрицы А равно 16, количество операций ввода элементов вектора В равно 4, количество операций вывода вектора С равно 4. Всего при последовательном выполнении этих операций ввода и вывода потребуется 24 момента времени.

1. систолическая обработка значительно снижает требования к вводу-выводу данных за счет их однократного ввода, последующего многократного использования и одновременной передачи по процессорной сети и однократного вывода.

Требования к вводу-выводу и передаче данных в СП составляет 14 тактов.

*Ускорение по вводу и выводу* элементов матрицы А, элементов вектора В, элементов вектора-результата *C*, достигаемое в процессорной сети, составляет:



где х -

В результате, таким образом, учитывая ускорение по операциям ввода и вывода, по своей длительности соответствующих операциям коммуникации, можно получить итоговое ускорение в процессорном поле.

1. существуют пессимистичные и оптимистичные оценки ускорения (рис. 67).

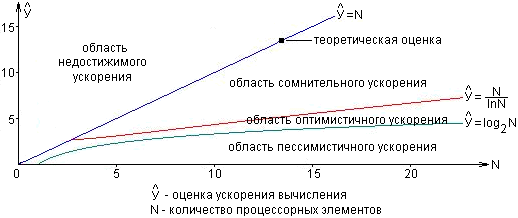


Рис. 112. Оценки ускорения вычисления

- рассчитанное *У* попало в область *пессимистичных ускорений* → результаты проектирования системы вполне могут быть улучшены (по ускорению);

- рассчитанное *У* попало в область *оптимистичных ускорений* → улучшить результаты скорее всего уже не удастся;

- рассчитанное *У* попало в область *сомнительных ускорений* → результаты необходимо тщательно проверить и тем самым найти допущенную где-то ошибку.

## Задача перемножения матриц

Объяснение способа умножения матриц из домашнего задания «Оценка прироста скорости перемножения матриц в многопроцессорных схемах»

В данном пункте рассматривается задача разработки специализированного вычислителя для умножения двух ленточных матриц, используемого при преобразовании координат. Цель работы - ускорить выполнение алгоритма программы.

Зададим операцию С = А \* В, где матрицы (4x4) имеют вид:



Построим для каждой матрицы матрицу, которая учитывает участие каждого элемента в вычислениях. Матрицы А и В - это пятидиагональная ленточная матрица (имеет 5 ненулевых диагоналей подряд); С - шестидиагональная ленточная матрица.

В качестве поддержки построения систолического поля сформулируем матрицу участия в вычислениях для каждого сомножителя aij, Ьij, а также матрицу с количеством

умножений для каждого элемента результата cij. Элемент а11 вводится один раз в

процессорное поле, используется в четырех вычислениях и прекращает свое существование. Элемент Ь11 вводится один раз в процессорное поле, используется в двух вычислениях и прекращает свое существование. Элемент c11 вводится один раз, используется два раза в вычислениях (операциях умножения) и один раз выводится. Вспомогательные матрицы создаются программно.

Выбираем локальную модель, задаем входные и выходные потоки и их направления:

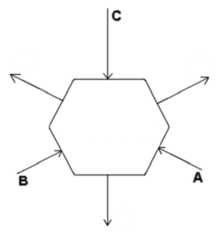


Рис. 113. Локальная модель ПЭ и направление потоков данных.

Шаг 1. Задали направления для трех потоков данных а11, Ь11, c11 (элементы главных диагоналей), отметили точками начало вычислений, ПЭ, в которых они используются.



Рис.114. Шаг 1.

Шаг 2. Через построенные точки на прямой а11 и с11 проводим прямые параллельные Ь11, при чем количество прямых над и под Ь11 соответствует количеству ненулевых диагоналей над и под главной диагональю соответственно в матрице В. По этим прямым будет перемещаться поток данных с элементами ленточной матрицы В.



Рис. 115. Шаг 2.

Шаг 3. Проводится еще одна прямая через точку прямой Ь11параллельно а11 .



Рис. 116. Шаг 3.

Шаг 4. Теперь надо разработать структуры двух входных и одного выходного потока данных.



Рис. 117. Шаг 4.

Пусть в выделенной точке (рис. 72) начинаются вычисления.

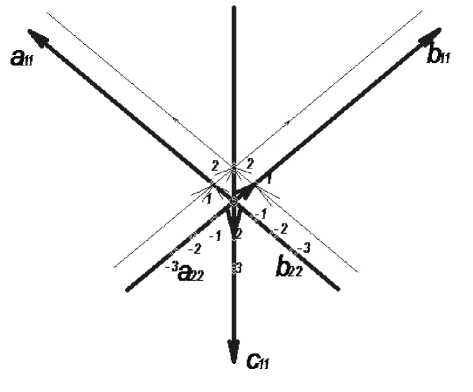


Рис. 118. Диаграмма следования элементов матриц в потоках данных.

Произведем замену узловых точек на последнем рисунке шестиугольными ПЭ. Спроектированное систолическое процессорное поле выполняет матричную операцию: C=A\*B с наложенными ограничениями на размерность и внутреннюю структуру матриц операндов (рис. 73).

Рис. 119. Процессорное поле

Правильность построения процессорного поля и можно проверить по вспомогательным матрицам. Время окончания работы систолического поля (время получения последнего элемента матрицы С) составляет t=15 тактов коммуникации и вычислений. Время получения элемента cij составляет: t( cij )=(5+i)+2(j-1).

Ускорение работы последовательного алгоритма осуществляется с помощью создания специального высокопроизводительного вычислительного блока. На основе разработанной структуры можно собрать заказную СБИС, в которой будет реализовано процессорное поле с 20 ПЭ с постоянными соединениями.

При каждом перемножение матриц будем использовать разработанное систолическое поле, которое будет являться спецвычислителем, обеспечивающим параллельную обработку данных вместо последовательного алгоритма выполнения программы. Таким образом, модернизируем последовательный алгоритм умножения матриц.