Цифровые процессоры обработки сигналов

Витязев С.В. РГРТУ. 2012

Раздел 3.1. Основы построения ЦСП — операционное ядро: вычислительные блоки и регистры.

В соответствии с рассмотренной ранее базовой (классической) архитектурой ЦСП основной ее составляющей является операционное ядро, включающее исполнительные элементы — вычислительные блоки — и управляющее ими устройство — программный автомат. Рассмотрим различные варианты построения операционного ядра ЦСП на примерах современных сигнальных процессоров. Примеры архитектур процессоров могут быть взяты из документации на процессор, называемой листом технических данных (Data sheet) и размещаемой, как правило, на сайте производителя микросхемы. В Приложении 1 будут приведены архитектуры наиболее характерных сигнальных процессоров с различными типами архитектур, представленные на современном рынке, выпущенные в различных годах фирмами Texas Instruments, Analog Devices и НПЦ «Элвис».

Начнем с рассмотрения вычислительных блоков.

Как уже было указано ранее, основными вычислительными блоками, состав которых вытекает непосредственно из основной нацеленности ЦСП на быструю реализацию операции умножения с накоплением, являются **умножитель**, **АЛУ** и **сдвигатель**.

Умножитель выполняет операцию умножения за один такт. Обычно он обозначается символом, изображенным на рис. 3.1. На каждом такте работы процессора умножитель принимает два операнда, подаваемые на его вход, перемножает их и выдает результат перемножения.

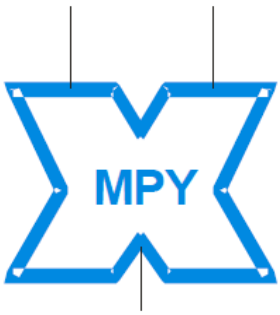

Рис. 3.1. Умножитель. Условное обозначение

Умножитель может выполняться как умножитель-накопитель (multiply-accumulate – MAC). В этом случае он самостоятельно выполняет умножение с накоплением за один такт, а блок АЛУ может быть параллельно задействован для выполнения других функций.

В некоторых процессорах результат на выходе умножителя появляется с задержкой на некоторое количество тактов. Операция умножения, в этом случае, все равно выполняется за один такт, только с задержкой (данная ситуация аналогична ситуации с обработкой данных в реальном масштабе времени).

Современные высокопроизводительные ЦСП используют обычно параллелизм обработки данных, включая в свой состав несколько умножителей. Другая современная тенденция модификации умножителя состоит в применении **параллелизма на уровне частей слова данных** (sub-word parallelism) [ ]. Поясним данное понятие. Когда мы говорим о том, что умножитель выполняет операцию умножения за один такт, мы имеем в виду перемножение двух слов данных (операндов) базовой длины. Например, если процессор является 16-разрядным, то все его внутренние шины для пересылки данных внутри процессора и все его вычислительные блоки рассчитаны на работу с 16-разрядными данными. Это базовая длина слова данных. Однако тот же умножитель за счет небольших модификаций может вместо перемножения двух 16-разрядных чисел выполнять за один такт две операции умножения 8-разрядных чисел, являющихся частями операндов, подаваемых на вход умножителя в обычном режиме. Идея параллелизма на уровне частей слов данных развита в настоящий момент до того, что современные ЦСП способны эффективно работать с 8-, 16-, 32- и 64-разрядными словами, то есть с той разрядностью, которой требует конкретное приложение. Кроме того, поддерживаются операции комплексного перемножения, когда действительная и мнимая части числа «упаковываются» в соответствующие части слов входных данных, и за один такт умножитель выполняет перемножение, сложение и вычитание требуемых частей двух комплексных чисел, формируя на выходе результат умножения в комплексной форме. Не стоит, однако, забывать, что если для конкретной задачи не требуется комплексная арифметика или применение вычислительных операций пониженной разрядности, то вся эта функциональность окажется невостребованной.

Арифметико-логическое устройство — АЛУ — применяется, в первую очередь, для выполнения за один такт операции сложения (вычитания), дополняя блок умножителя в выполнении умножения с накоплением. Обозначается блок АЛУ, как правило, символом рис. 3.2. В то же время, если блок умножителя является основным с точки зрения реализации базовой для алгоритмов ЦОС операции умножения-накопления, блок АЛУ является наиболее многофункциональным. На него возлагается множество дополнительных функций. Именно поэтому он называется не просто сумматором, а блоком арифметико-логических

операций. К данным операциям кроме сложения/вычитания относятся логические операции И, ИЛИ, НЕ и другие, выполняемые над целыми словами данных или между каждой парой бит двух операндов. Например, если на входе блока АЛУ имеются операнды 10 и 01 в двоичной системе счисления, то команда логического И даст на выходе АЛУ результат 01. Если же требуется выполнить операцию побитового И, то это также возлагается на блок АЛУ и дает результат 00.

Рис. 3.2. АЛУ. Условное обозначение

Блок АЛУ обычно участвует в операциях сравнения. Два операнда на входе АЛУ могут быть проверены на условия «больше», «меньше», «равно» и другие. Такие операции широко распространены в программировании и, в частности, в задачах обработки сигналов и реализуются блоком АЛУ.

Другим типом операций, часто выполняемых АЛУ, являются специальные команды, расширяющие возможности процессора. Например, вычисление обратной величины, вычисление квадратного корня, другие операции. Процессор включает в свой состав специальную аппаратную поддержку таких операций, расширяющую возможности АЛУ.

Также как и для умножителя, для блока АЛУ характерно дублирование и поддержка параллелизма на уровне частей слова данных. В частности, интересным примером является возможность современных блоков АЛУ вычислять за один такт сумму и разность двух входных операндов. Эта возможность широко востребована алгоритмом БПФ, делая его реализацию существенно эффективнее.

Сдвигатель (рис. 3.3) выполняет основную функцию масштабирования результатов вычислений путем сдвига бит двоичного операнда вправо или влево на требуемое число разрядов. Напомним, что сдвиг двоичного числа вправо на один разряд соответствует делению числа на 2. Перемножение двух чисел дает в общем случае расширение разрядной сетки вдвое. Таким образом, после каж-

дой операции умножения необходимо произвести сдвиг результата вправо на число разрядов, соответствующее базовой длине слова данных процессора. Точность результата при этом существенно падает, но числа остаются в пределах разрядной сетки.

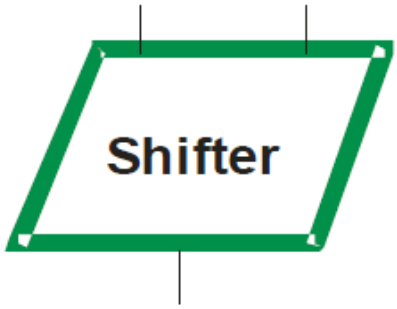

Рис. 3.3. Сдвигатель. Условное обозначение

Сдвигатель способен выполнять арифметический и логический сдвиг. Арифметический сдвиг означает сдвиг с учетом знака. Знаковым битом, как правило, является старший разряд в двоичном представлении числа. Естественно, масштабирование не должно затрагивать знакового бита. Особенно это важно при сдвиге бит влево, когда знак числа может быть изменен. Сдвиг числа с сохранением его знака называется арифметическим сдвигом. Логический сдвиг смещает биты безотносительно к знаковому разряду.

При сдвиге бит вправо или влево возникает вопрос, чем заполнять освобождающиеся разряды? В связи с этим различают сдвиг с заполнением нулями, с заполнением единицами, с расширением знаком (биты принимают значение знакового бита) и другие. Отдельно выделим циклический сдвиг, при котором освобождающиеся разряды заполняются значениями бит, вытесняемых из разрядной сетки. Например, если осуществляется циклический сдвиг двоичного числа 111000111 вправо на два разряда, то результатом операции окажется число 111110001.

Кроме операций сдвига блок сдвигателя широко применяется для реализации других типов побитовых операций, включая побитовое И, ИЛИ, НЕ и другие. В этом он помогает блоку АЛУ. Кроме того, в некоторых моделях процессоров блок сдвигателя применяется и для простых арифметических действий, таких как целочисленное сложение/вычитание, а также для реализации команд перехода с одного с текущей строки программного кода на следующую строку, где он является «помощником» программного автомата.

К сдвигателю также относится сказанное ранее о дублировании вычислительных блоков и параллелизме на уровне частей слов данных.

Генераторы адреса данных (рис. 3.4) — выполняют функцию формирования адресов операндов, которые требуется извлечь из памяти данных и подать на вход одного из вычислительных блоков. Представим, что фрагмент входного сигнала (последовательность отсчетов), необходимый для выполнения некоторого преобразования, записан в память данных процессора. В соответствии с заданным алгоритмом обработки требуется извлекать отсчеты сигнала из памяти в определенном порядке и подавать их на вычислительные блоки. Таким образом, становятся необходимыми устройства, отвечающие за генерацию запросов отсчетов, хранящихся в памяти, в требуемом порядке. Генерация запросов называется формированием адресов данных, а блоки ее реализующие – генераторами адреса данных.

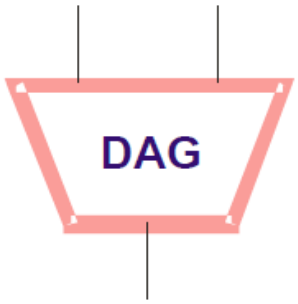

Рис. 3.4. Генератор адреса данных. Условное обозначение

Память соединяется с операционным ядром набором шин адреса и данных. По шине адреса к памяти поступает запрос содержимого требуемой в настоящий момент ячейки. В ответ на этот запрос на шину данных выкладывается соответствующее значение.

Каким образом генератор адреса данных формирует адреса? Существуют различные принципы генерации адреса. Они называются режимами адресации и будут рассмотрены ниже при описании архитектур построения памяти ЦСП. Отметим лишь, что простейшим вариантом генерации адреса является выделение его из тела команды (непосредственная адресация), а одним из наиболее сложных вариантов может считаться циклическая адресация с модификацией адреса, для которой требуется выполнение набора арифметических действий.

Последнее утверждение особенно важно для нас в рамках данного раздела. Оно позволяет пояснить, что, поскольку генератор адреса способен выполнять несложные математические действия, то его можно использовать не только для ге-

нерации адресов, но и для реализации арифметики, усиливая блоки АЛУ. Данный факт широко применяется в современных ЦСП. Например, в процессорах TigerSHARC фирмы Analog Devices вместо генераторов адреса данных используется термин целочисленное АЛУ, что отражает выше сказанный подход к построению ядер ЦСП.

Для генераторов адресов данных, как и для остальных вычислительных блоков, справедлив принцип дублирования и работы с частями слов данных.

Регистры (рис. 3.5) — представляют собой быструю локальную память, рассчитанную на хранение одного слова данных (операнда). Поясним это. Как уже было сказано, вычислительные блоки являются основными компонентами ядра, непосредственно выполняющими арифметические действия над данными — отсчетами обрабатываемого сигнала. Отсчеты сигнала, как было показано при описании общих принципов функционирования ЦСП, поступая на вход процессора, запоминаются во внутренней памяти данных. Передача данных между памятью процессора и вычислительными блоками, как указывалось при описании работы генераторов адресов данных, осуществляется посредством набора шин адреса и данных. Если бы входные операнды каждого из вычислительных блоков и результаты вычислений на каждом такте работы выбирались из памяти и направлялись в память, потребовалось бы наличие слишком большого набора шин. В то же время, в алгоритмах обработки сигнала очень часто один и тот же коэффициент преобразования участвует в арифметических действиях с большим числом отсчетов сигнала. Нет смысла каждый раз брать его из памяти. Достаточно загрузить его однократно. Аналогично результат повторяющихся операций умножения с накоплением необходимо направлять в память только после завершения всех накоплений. Когда умножитель получил результат для очередной пары чисел, этот результат следует подать на сумматор — АЛУ — для накопления произведений. Нет смысла записывать результат умножения в память, чтобы тут же вновь считать его. Генераторы адреса данных используют для работы с сигналом адрес массива его отсчетов, записанных в память процессора. Генератор работает с этим адресом на протяжении всего заданного фрагмента кода, переходя от одного отсчета к другому с использованием разных режимов адресации, включая циклическую и бит-реверсную. Для генератора адреса данных целесообразно все это время «иметь под рукой» адрес начала массива, с которым идет работа.

Все перечисленное приводит нас к тому, что для эффективной работы операционного ядра процессора, вычислительные блоки должны иметь собственную локальную память, куда они могли бы записывать те операнды или результаты вычислений, которые могут им еще понадобиться в скором времени. Такую локальную память и представляют собой регистры. Таким образом, регистры используются для хранения операндов, подаваемых на вход арифметических бло-

ков, генераторов адреса данных, в качестве аккумуляторов результатов и для размещения результатов вычислений.

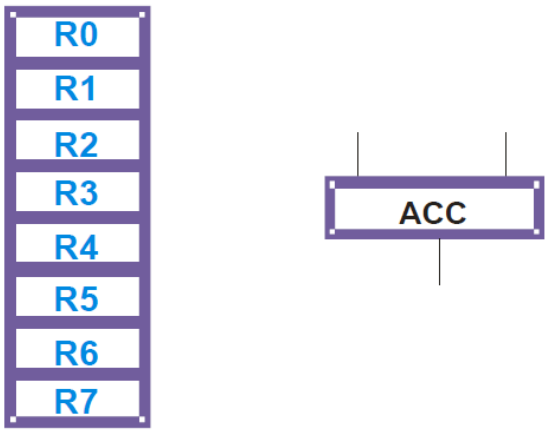

Рис. 3.5. Регистры. Условное обозначение

Современные тенденции построения процессоров ориентированы на использование языков программирования высокого уровня, когда программа для процессора пишется, скажем, на языке Си, а ее перевод на язык архитектуры конкретного ЦСП выполняется автоматически средствами генерации кода. Язык Си является универсальным и не учитывает особенности архитектуры сигнальных процессоров. В связи с этим, хороший перевод программного кода с максимальным задействованием всех возможностей архитектуры сделать автоматически оказывается достаточно сложно. Существенно упросить данную задачу позволяет принцип построения архитектур процессоров, именуемый ”Load-Store”. В соответствии с ним все операнды, подаваемые на вычислительные блоки, и результаты вычислений проходят через регистры. Это упрощает набор команд, делает их однотипными и универсальными. Для компилятора языка высокого уровня данный подход оказывается очень важен.

В современных процессорах регистры объединяют в **регистровые файлы** — группы регистров, взаимодействующих с вычислительными блоками. При этом стараются не привязывать к конкретному блоку только определенные регистры, а делать регистры универсальными. Минимальное количество ограничений на использование регистров, возможность доступа любым вычислительным блоком к любому регистру регистрового файла и большое число регистров повышают гибкость и эффективность архитектур ЦСП. Поддержка регистрами возможности работы с частями слов данных также является современной тенденцией построения ЦСП

Кроме регистров общего назначения, используемых вычислительными блоками, в состав ЦСП входят специальные регистры. В первую очередь, это регистры управления режимами работы ЦСП и регистры текущего состояния процессора.

К другому типу регистров относятся регистры, отображенные в памяти. Такие регистры не являются физически автономными компонентами процессора. Они представляют собой ячейки с определенными адресами, выделенные в общем пространстве внутренней памяти ЦСП для определенных задач. Выделять регистры в физически автономные компоненты имеет смысл, только если с этими регистрами важно работать с высокой скоростью и параллельно с другими действиями.

В заключение отметим следующее. Можно выделить два основных пути повышения быстродействия процессоров. Первое направление – это повышение тактовой частоты. Тактовая частота задает скорость работы всех элементарных блоков процессора и при том же составе архитектуры ЦСП может приводить к соответствующему повышению производительности. Повышение тактовых частот ЦСП связано с технологией изготовления кристаллов и ограничивается в основном физическими возможностями современных технологий. В то же время архитектурно процессоры также должны быть способны поддерживать высокую рабочую частоту. Примером может являться командный конвейер, о котором речь пойдет ниже.

Второе направление повышения быстродействия процессоров связано с переходом от того, чтобы быстрее делать каждое действие, к тому, чтобы за то же время сделать большее число действий, то есть с распараллеливанием операций.

Предположим, алгоритм обработки сигнала требует выполнения 100 операций умножения с накоплением. Пусть процессор реализует одну операцию MAC за такт. Чем меньше длительность такта (то есть чем выше тактовая частота), тем быстрее процессор справится с поставленной задачей. В то же время, если процессор способен выполнять 2, 3 или 4 операции MAC за такт, то при той же тактовой частоте это даст соответствующее повышение быстродействия. Чтобы процессор был способен выполнять несколько умножений с накоплением за такт, в его состав должно входить соответствующее количество умножителей или один умножитель должен быть способен выполнять одновременно несколько операций MAC.

Распараллеливание может реализовываться на уровне частей слов данных, о чем было рассказано выше. Распараллеливание может вестись на уровне вычислительных блоков, когда состав блоков дублируется. И, наконец, распараллеливание может быть реализовано на уровне ядер. Здесь мы приходим к многоядерным процессорам, представляющим собой одно из наиболее актуальных направлений развития процессорной техники.

Распараллеливание, кроме повышения быстродействия, дает еще один плюс. Оно дает меньший уровень повышения энергопотребления по сравнению с наращиванием тактовой частоты. Реализация того же количества операций с меньшей тактовой частотой требует меньшего энергопотребления. В современных условиях энергоэффективность ЦСП начинает играть особо важную роль.

Здесь вновь следует отметить, что высокопараллельные, в том числе многоядерные, процессоры требуют распараллеливания алгоритма обработки сигнала. Если алгоритм не может быть разбит на фрагменты, исполняемые параллельно, вся вычислительная мощность процессора оказывается невостребованной.

Контрольные вопросы:

1. Что входит в состав операционного ядра?
2. Какие функции выполняет умножитель?
3. Что такое параллелизм на уровне частей слова данных? Чего он позволяет достичь?
4. Всегда ли позволяет распараллеливание операций, например, на уровне частей слов данных увеличить быстродействие процессора?
5. Какие уровни распараллеливания операций вы можете назвать?
6. Какие функции выполняет АЛУ?
7. Что такое арифметический и логический сдвиги?
8. Запишите результат логической и побитовой логической операции И для следующих двух чисел, представленных в двоичной системе счисления: 1011 и 0110.
9. Какие функции выполняет сдвигатель?
10. Почему операция сдвига является необходимой в процессе вычислений?
11. Какие функции выполняют генераторы адреса данных?
12. Зачем нужна шина адреса? Зачем нужна шина данных?
13. Каково назначение регистров в составе операционного ядра ЦСП?
14. Что такое регистровый файл?