**Вопросы построения специализированных вычислительных систем обработки данных на FPGA с использованием софт-процессорных ядер (вне зависимости от архитектуры)**

Общеизвестные факты, которые, так или иначе, всплывают в спорах - что применять в проекте – микроконтроллер, или ПЛИС/FPGA.

При любом практически раскладе проекты на базе ПЛИС оказываются дороже и медленнее решений на базе стандартных микросхем/микроконтроллеров или специализированных микросхем (ASIC, ASSP — Application-Specific Integrated Circuit, Application-Specific Standard Product). Но это только в том случае, если речь идёт о массовой продукции. В случае сравнения с микроконтроллерами сравнение по чистой производительности для стандартных решений (ядер) будет также не в пользу софт-процессора (редко от него можно ожидать тактовой частоты более 50-150МГц). Программируемые связи между элементами FPGA и блоками логики вносят заметно большие задержки, чем металлизированные связи внутри микросхем с фиксированной структурой.

Сравнение по функциональности микроконтроллерные решения будут выигрывать лишь в «топовом» варианте – их версии с большим количеством линий ввода-вывода и множеством различных периферийных устройств. И то вопрос цены решения будет сопоставим с FPGA – цена на микроконтроллеры в таком исполнении и комплектации может быть существенна, плюс возникает вопрос стоимости разработки программного обеспечения, способного корректно управлять множеством устройств. Да, конечно, есть ОСРВ которая может решить ряд проблем, но это потеря производительности, плюс любой микроконтроллер имеет ограничения по возможностям одновременного использования своих периферийных устройств, как минимум из-за совместного использования внешних выводов.

Ключевыми преимуществами ПЛИС/FPGA являются возможность создания в них параллельных вычислительных структур, реализации любых периферийных интерфейсов или цифровых исполнительных узлов практически без ограничений по количеству. Эти свойства проистекают из возможности реализовать в ПЛИС множество небольших блоков, реализующих простые операции. ПЛИС с архитектурой FPGA традиционно используются в задачах цифровой обработки сигналов, поскольку обеспечивают очень высокие показатели производительности. Важным и существенным сегментом применения FPGA являются телекоммуникационные устройства, - магистральное сетевое оборудование, базовые станции.

Однако на практике единственной нишей, где FPGA не подходят в качестве аппаратной платформы, являются микропотребляющие устройства с аккумуляторным питанием.

Наличие в современных FPGA множества аппаратных узлов DSP, блочной памяти в сочетании с возможностью объединения их в практически любые конфигурации позволяет использовать в проектах преимущества аппаратного параллелизма при выполнении операций. Также отдельные части алгоритмов могут быть также реализованы в программируемой логике. Это касается также манипуляцией, например и с блоками данных в памяти и работы с числами большой разрядностью (бывает важно при работе с алгоритмами ЦОС).

Одна из ниш для ПЛИС/FPGA, особенно в сегменте бюджетных низко-, среднескоростных серий – преобразователи интерфейсов, или системы с множественными интерфейсами. Это возможно из-за того, что в большинстве микросхемы FPGA выпускаются в корпусах с большим количеством внешних выводов, плюс возможность сконфигурировать логику под любое мыслимое количество периферийных интерфейсов.

Системы сбора и предобработки данных, а также системы управления также является достаточно распространенной задачей для система на базе FPGA. Прежде всего возможность параллельной обработки/опроса возможно разномастных (с различными типами интерфейсов) датчиков. А чаще даже – датчиков с однотипными интерфейсами – например, UART /I2C/SPI в количестве, превышающем типичное количество таких интерфейсов у микроконтроллера. При этом изменения требований к проекту – количество и набор периферийных устройств, скорости передачи данных, порядок опроса при этом оказывают уже не существенное влияние и будут выражены только в изменении внутренней логики.

Аналогичные требования могут предъявляться и к системам управления практически это связка систем сбора данных и управления исполнительными элементами. Для реализации системы управления требуется не только организация основного цикла регулирования, но и обработка реакций на исключительные ситуации, и желательно в масштабе реального времени.

Следует всегда помнить, что софт-процессоры в проектах на FPGA не должны быть основным компонентом, определяющим вычислительную мощь и производительность. Основное предназначение софт-процессоров в предоставлении разработчику возможности управление другими компонентами проекта. В том числе одной из задач софт-процессоров - перенести часть процесса отладки на программные инструменты – изменение программы управления, алгоритма взаимодействия компонентов FPGA проекта или иных настроек будет произведено много быстрее, чем пересборка всего HDL-проекта. Не стоит исключать также и возможность интерактивной работы.

Практика применения микропроцессоров и микроконтроллеров подразумевает, что основную вычислительную нагрузку в системе несет процессорное ядро. Также при этом процессорное ядро обрабатывает прерывания от различных периферийных устройств. Несколько снижается нагрузка на ядро в том случае, если периферийные устройства имеют прямой доступ к памяти системы.

Отголоски этой практики можно наблюдать в ряде софт-процессоров, иногда содержащих элементы подсистем традиционных микроконтроллерных систем, например, многоуровневые контроллеры прерываний, системные шины и т.п. Да, безусловно такой подход понятен – он как минимум позволяет перебросить ряд задач на плечи программистов микроконтроллеров, предоставив им знакомую и понятную инфраструктуру и позволив им использовать привычные среды разработки, библиотеки и наработки. Но….Такой подход не отвечает «духу FPGA».

При разработке систем на базе FPGA добавление софт-процессора стоит рассматривать лишь как дополнительный элемент, упрощающий динамическую настройку всей системы и упрощающий отладку и развитие проекта в целом. Более того – софт-процессорное ядро может интегрироваться с основными исполнительными узлами проекта любым удобным способом! Не обязательно копируя решения встречающиеся в обычных микроконтроллерах – в случае с реконфигурируемой структурой реализуемой логики в FPGA это совершенно не обязательно. Не исключён также вариант, когда в FPGA проекте может быть задействовано несколько относительно простых ядер.В случае с архитектурой RISC-V это могут быть простые ядра с поддержкой базовой системы команд – RV32I/RV32IE/ RV32IEC.

Планирование архитектуры на ранней стадии проектирования может существенно облегчить разработку программного обеспечения, особенно в части обработки независимых событий, которые в реальности могут перекрываться. Для конфигурируемой системы, разрабатываемой для конкретного набора периферийных устройств, протоколов и интенсивности внешних воздействий, вполне может использоваться набор из нескольких ядер RV. Такой подход может показаться странным и не рациональным, но – взятая по отдельности программа для управления каждым из независимых ядер будет заметно проще, в том числе и в отладке.

Возможности архитектуры RISC-V по расширению/дополнению системы команд позволяют органично «вписать» взаимодействие софт-ядра с основными исполнительными или вычислительными ресурсами FPGA-проекта.

Именно возможность организации массового параллелизма при вычислениях и работе является важным и принципиальным конкурентным преимуществом FPGA как аппаратной платформы для проектов. При грамотном планировании и применении ресурсов FPGA способны обеспечить за счет параллельной работы не только высокую абсолютную производительность, но и соотношение производительность/цена, сопоставимое, или даже превосходящее это соотношение для сигнальных процессоров и процессоров/микроконтроллеров общего назначения.

Недостаток в абсолютном значении тактовой частоты софт-процессора можно компенсировать архитектурными решениями. Так, при работе с обычными микропроцессорными системами разработчик не рассматривает память в качестве отдельного ресурса, поскольку все используемые данные автоматически помещаются в основную память. При оптимизации производительности при этом можно рассуждать о работе с регистрами, кэшировании, работе с прямым доступом в память и т. п.

В случае с FPGA возможности работы данных много больше.

В современных FPGA (даже «бюджетных», «не топовых» серий) память может быть реализована через:

• триггеры логических ячеек;

• LUT секции (распределенная память);

• аппаратными блоками памяти (BRAM).

(Можно также иметь в виду и внешнюю память, как правило, динамическую, которая может иметься на плате и для доступа к которой может быть реализован контроллер (как правило, IP-ядра контроллеров доступны в библиотеках среды разработки производителя FPGA)).

Важным преимуществом FPGA является крайне высокая теоретическая пропускная способность подсистемы памяти в целом. Память BRAM представляет собой обычную статическую память (это именно аппаратный блок, следовательно, внутри него соединения металлические и с минимальными задержками). Блоки имеют полностью независимые интерфейсы и могут работать на системной тактовой частоте. Таким образом, теоретически, FPGA большого объема способны обеспечить весьма высокую пропускную способность.

Однако важнейшим вопросом при этом является то, способен ли проект получить преимущество от множества параллельно работающих BRAM, и может ли вообще быть обеспечен такой поток данных внутри кристалла.

Еще следует отметить тот момент, что блоки BRAM можно объединять между собой и конфигурировать такие объединения на любую мыслимую разрядность (в рамках ресурсов конкретной FPGA, конечно) – от одного бита до нескольких тысяч. И опять же, пользуясь адаптивностью RISC-V можно добавить в софт-процессор поддержку битовых, векторных или SIMD расширений.

При этом никто не запрещает разделять такие специальные области памяти с пространством памяти софт-процессора (вспомним, одной из «фишек» процессоров Apple M\* является единое пространство основной и видео-памяти – и этот «трюк» сработал!).

Обычно при работе с микропроцессорными системами программист, особенно работающий на языках высокого уровня, не заботится вопросами распределения памяти, переменных, массивов и пр. Исключением могут быть редкие случаи многоядерных систем, имеющих специфическую структуру памяти, систем доступа к ней или топологии связи между ядрами.

Чтобы эффективно работать с памятью в FPGA, нужно найти оптимальное решение для управления высокой суммарной пропускной способностью интерфейсов отдельных блоков памяти. Если все данные будут храниться в одном блоке памяти ради экономии ресурсов, его производительность будет ограничена двумя операциями за такт (так как блочная память в FPGA имеет два порта). Однако если разделить программные объекты между разными блоками памяти, каждый из них сможет выполнять две операции за один такт. Как и в случае с DSP-блоками, задача заключается в повышении количества операций за такт благодаря использованию большего объёма ресурсов FPGA в проекте.

Итого:

Интеграция софт-процессорных ядер в FPGA проект не самоцель и не средство увеличения производительности;

Софт-процессор не основной вычислитель, а лишь инструмент управления и облегчения общей работы с системой;

Интеграция периферийных систем возможна и в адресное пространство софт-процессора, и через любые системные шины.

В рамках архитектуры RISC-V взаимодействие с внешними устройствами возможно или как с элементами памяти, или через работу с регистрами специальных функций. Обработку данных при необходимости можно оптимизировать через собственное, оптимизированное для задачи расширение системы команд.