Улучшение характеристик ISR

Если ваша программа интенсивно использует аппаратные прерывания, характеристики ISR являются, вероятно, самым значимым фактором из всех ваших программных характеристик. В этой секции обсуждаются аппаратные и программные стратегии улучшения характеристик ISR.

Программное улучшение характеристик

Для улучшения характеристик вашего ISR, вы, вероятно, сначала рассматриваете изменения в программе. Однако, в большинстве случаев, могут потребоваться изменения в аппаратной части, которые улучшат эффективность системы. Обсуждения аппаратной оптимизации в секции "Аппаратное улучшение характеристик" на стр. 8-24.

В следующих секциях описываются изменения, которые вы делаете в программном проекте для улучшения характеристик ISR.

Исполнение время затратных алгоритмов в контексте приложения

ISR предлагает быстрый, с малой задержкой отклик на изменения в состоянии аппаратной части. Ей потребуется минимум усилий для сброса аппаратного прерывания и возврата. Если ISR выполняется долгие, некритичные процессы, она может столкнуться с большим количеством важных задач в системе.

Если вашей ISR требуется длительное исполнение, сделайте в вашей программе исполнение этих процессов снаружи контекста исключения. ISR может использовать механизм передачи сообщений для извещения кода приложения о выполнении задач длительных процессов.

Откладывать задачи очень просто в системах, основанных на RTOS, таких как MicroC/OS-II. В этом случае, вы можете создавать поток для обработки операций, требующих интенсивной загрузки процессора, а ISR может быть связана с этим потоком, используя любой механизм связи RTOS, такой как флаг события или очередь сообщения.

Вы можете эмулировать этот подход для однопоточной системы, основанной на HAL. Основная программа опрашивает глобальную переменную, управляемую ISR, для определения необходимости выполнения операций, требующих интенсивной загрузки процессора.

Аппаратная реализация время затратных алгоритмов

Задачи, требующие интенсивной загрузки процессора, зачастую должны передавать большие массивы данных к и от периферии. Основной процессор, такой как Nios II, - не самый эффективный способ сделать это. Используйте аппаратные средства прямого доступа к памяти (DMA), если они доступны.

За дополнительной информацией о программировании аппаратных средств DMA, обратитесь к секции "Использование DMA устройств" в главе "Разработка программ с использованием слоя аппаратной абстракции" в настольной книге программиста Nios II.

Увеличение размера буфера

Если вы используете DMA для трансферта длинных буферов данных, размер буфера может влиять на характеристики. Малые буферы подразумевают частые прерывания, которые приводят к увеличению издержек. Увеличьте размер буферов транзакций данных.

Использование дублирующей буферизации

Использование DMA для трансферта длинных буферов данных может не дать существенного улучшения характеристик, если процессор Nios II должен будет ожидать окончания DMA транзакций, прежде чем выполнить следующую задачу.

Дублирующая буферизация позволяет процессору Nios II обрабатывать один буфер данных, когда аппаратная часть передаёт данные в или от другого.

Разрешённое удержание прерываний

Когда прерывания запрещены, процессор Nios II не может быстро ответить на событие аппаратного прерывания. Буферы и очередь могут быть заполнены или переполнены. Даже в отсутствии переполнения, максимальное время обработки прерывания может увеличится после повторного разрешения прерываний, поскольку ISR необходимо обработать незавершённые данные.

Запрещайте прерывания реже, насколько возможно, и на минимально возможное время. Взамен запрещения всех прерываний, вызывайте alt_ic_irq_disable() и alt_ic_irq_enable() для запрещения и разрешения конкретных прерываний.

Для защиты структур общих данных, используйте структуры RTOS, такие как семафоры. Запрещайте все прерывания только для важных системных операций. В коде во время запрета прерываний, выполняйте только минимальное количество важных операций и сразу же разрешайте прерывания.

Использование быстрой памяти

Характеристики ISR зависят от скорости памяти.

Для улучшения характеристик размещайте ISR и стек в самой быстрой памяти: предпочтительно в сопряжённой памяти (если такая есть) или внутри чиповой памяти.

Если не возможно разместить основной стек в быстрой памяти, рассмотрите использование отдельных стеков исключений, распределённых в секциях быстрой памяти, как будет описано в следующей секции.

За информацией о распределении памяти, обратитесь к секции "Использование памяти" в главе "Разработка программ с использованием слоя аппаратной абстракции" в настольной книге программиста Nios II.

За информацией о сопряжённой памяти, обратитесь к главе "Кэш и сопряжённая память" в настольной книге программиста Nios II.

Использование отдельных стеков исключений

HAL реализует два типа отдельных стеков исключений. Их доступность зависит от контроллера прерываний, как описано в этой секции. В табл. 8-3 отображена доступность отдельных стеков исключений и то, как они могут использоваться с каждым типом контроллера прерываний.

Использование отдельных стеков исключений влечёт за собой дополнительные издержки. Когда обрабатывается программное исключение или аппаратное прерывание, процессор должен исполнять дополнительную инструкцию, на вход и на выход, для смены указателя стека. Учитывайте это время на дополнительные процессы, если вам требуется точно вычислять время отклика на прерывание.

Общий отдельный стек исключений

Общий отдельный стек исключений доступен с внешним и с внутренним контроллером прерываний.

Используйте настройку BSP hal.linker.enable_exception_stack для разрешения общего отдельного стека исключений.

Код общего направителя (funnel) исключений HAL следит за корректным изменением указателя стека на входе и на выходе из обработчика исключений.

Аппаратный отдельный стек исключений

Аппаратный отдельный стек исключений доступен с интерфейсом EIC. Аппаратный отдельный стек исключений не применяется с IIC. Аппаратные прерывания и программные исключения в IIC используют один стек.

Следующие BSP настройки разрешают вам контролировать аппаратный отдельный стек исключений:

- hal.linker.enable_interrupt_stack – разрешает аппаратный отдельный стек исключений.
- hal.linker.interrupt_stack_size – контролирует размер аппаратного отдельного стека исключений.
- hal.linker.interrupt_stack_memory_region_name – разрешает вам контролировать место размещения стека аппаратных прерываний в памяти.

Код общего направителя (funnel) исключений HAL следит за корректным изменением указателя стека на входе и на выходе ISR.

Табл. 8-3. Использование отдельных стеков исключений

Контроллер прерываний	BSP настройки		Стек приложения	Общий стек исключенй	Стек аппаратных прерываний
	Общий отдельный стек исключений	Аппаратный отдельный стек исключений			
Внутренний	Нет	-	▪ Приложение ▪ Программные исключения ▪ Аппаратные прерывания	-	-
	Да	-	▪ Приложение	▪ Программные исключения ▪ Аппаратные прерывания	-
Внешний	Нет	Нет	▪ Приложение ▪ Программные исключения ▪ Аппаратные прерывания	-	-
		Да	▪ Приложение ▪ Программные исключения	-	▪ Аппаратные прерывания
	Да	Нет	▪ Приложение ▪ Аппаратные прерывания	▪ Программные исключения	-
		Да	▪ Приложение	▪ Программные исключения	▪ Аппаратные прерывания

Если ваша ISR расположена в таблице векторов, HAL не будет предоставлять код направителя. В этом случае, ваш код должен управлять указателем стека, на манер любой другой функции кода направителя.

За подробной информацией о реализации отдельных стеков аппаратных прерываний, обратитесь к *AN595:* *Vectored Interrupt Controller Applications and Usage*.

Использование вложенных аппаратных прерываний

По умолчанию, HAL запрещает прерывания, когда координирует ISR. Это означает, что одновременно может исполняться только одна ISR, а ISR исполняются по принципу первый пришёл – первый обслужен. Это уменьшает избыточность системы, ассоциированную с обработкой прерываний, и упрощает разработку ISR. ISR не требуют повторного использования. ISR могут использовать и модифицировать любые глобальные или статичные структуры данных или аппаратные регистры, которые не обобщены с кодом приложения.

Однако принцип первый пришёл - первый обслужен принуждает HAL установить приоритет аппаратных прерываний в случае, когда два IRQ активны одновременно. Прерывание низкого приоритета, случившееся перед прерыванием высокого приоритета, может препятствовать исполнению высоко приоритетной ISR. Это форма инвертирования приоритета, она может иметь значительное влияние на характеристики ISR в системах, которые генерируют частые прерывания.

Программная среда может обеспечить полное назначение приоритетов всех аппаратных прерываний, используя вложенные ISR. Во вложенных ISR высоко приоритетные прерывания разрешены в низкоуровневых ISR.

Такая технология может улучшить время отклика для высокоприоритетных прерываний.

Вложенные ISR увеличивают время обработки низкоуровневых аппаратных прерываний.

Если ваша ISR очень короткая, ей не требуется сервис разрешения вложенных аппаратных прерываний. Разрешение вложенных прерываний для короткой ISR может значительно увеличить время отклика для высокоуровневых прерываний.

Если вы используете отдельный стек исключений в IIC, вы не сможете вложить аппаратные прерывания. За информацией об отдельном стеке исключений, обратитесь к секции "Использование отдельных стеков исключений".

Вложенные аппаратные прерывания и внутренний контроллер прерываний (IIC)

Для реализации внутренних аппаратных прерываний с IIC, используйте функции alt_irq_interruptible() и alt_irq_non_interruptible() для заключения кода в скобки в ISR, требующей интенсивной работы процессора. Вызов alt_irq_interruptible() устанавливает маску прерывания таким образом, чтобы высоко приоритетные прерывания забирали контроль от запущенной ISR. Когда ваша ISR вызывает alt_irq_non_interruptible(), маска прерывания возвращается в предыдущее состояние.

Если ваша ISR вызывала alt_irq_interruptible(), она должна вызвать alt_irq_non_interruptible() перед возобновлением работы. Иначе система обработки исключений HAL может зациклится.

Вложенные аппаратные прерывания и внешний контроллер прерываний (EIC)

Расширенные HAL API прерываний поддерживают вложенные аппаратные прерывания, также известные как обслуживание приоритетных прерываний. Драйвер устройства должен быть написан специальным образом, чтобы корректно функционировать в условиях приоритетов.

Устаревшие драйверы устройств не распространяются о свойствах isr_preemption_supported. Поэтому SBT думает, что они не поддерживают приоритеты. Если ваш собственный устаревший драйвер поддерживает приоритеты, и вы хотите разрешить приоритеты в вашем BSP, вы должны усовершенствовать драйвер для использования с расширенной HAL API прерываний.

За подробной информацией о приоритетах в EIC обратитесь к секции "Управление приоритетом" на стр. 8-13.

В таблице векторов, HAL вставляет скобки для корректировки направителей для каждого аппаратного прерывания, в зависимости от настроек приоритета.

Размещение тела ISR в таблице векторов

Если вы используете векторный EIC, и у вас есть критичная ISR малого размера, вы можете добиться улучшения рабочих характеристик, разместив код ISR прямо в таблице векторов. В этом случае, вы исключите избыточность ветвлений из таблицы векторов посредством направителя HAL на вашу ISR.

Драйвер EIC размер входных данных таблицы векторов по умолчанию. Например, для Altera VIC размер по умолчанию -16 байт. Для подгонки вашей ISR, отрегулируйте размер входных данных в настройках драйвера, на стадии создания BSP.

Размещение ISR в таблице векторов – это расширенная и подверженная ошибкам технология, не поддерживаемая явно HAL. Вы должны подходить к ней с большой осторожностью, проверяя, как код ISR заполняет таблицу векторов. Если ваш код ISR переполняет таблицу векторов, он может повредить другие элементы в таблице векторов, и даже всю систему обработки прерываний.

Когда ваша ISR размещена в таблице векторов, её не нужно регистрировать. Не вызывайте функцию alt_ic_isr_register(), поскольку она перепишет содержимое таблицы векторов. HAL не может предоставить код направителя (funnel). Поэтому, вы должны будете управлять всеми кодами направителей.

За подробной информацией о размещении ISR в таблице векторов, обратитесь к *AN595: Vectored Interrupt Controller Applications and Usage*.

Использование оптимизации компилятора

Для улучшения характеристик в контексте исключений и контексте приложений, используйте уровень оптимизации компилятора -O3. Уровень -O2 также даёт хорошие результаты. Удаление оптимизации всегда значительно увеличивает время отклика исключения.

За подробной информацией об оптимизации компилятора, обратитесь к секции "Уменьшение размера кода" в главе "Разработка программ с использованием слоя аппаратной абстракции" в настольной книге программиста Nios II.