В.С. Волин, Е.М. Кравцунов, д.т.н., проф. С.В. Семенихин, д.т.н. В.М. Фельдман, С.А. Черепанов (ОАО «ИНЭУМ им. И.С. Брука», ЗАО «МЦСТ»)

V. Volin, E. Kravtsunov, S. Semenikhin, V. Feldman, S. Cherepanov

УПРАВЛЕНИЕ ЭНЕРГОПОТРЕБЛЕНИЕМ ПРОЦЕССОРНЫХ ЯДЕР ИЗ ОПЕРАЦИОННОЙ СИСТЕМЫ ДЛЯ ПРОТОТИПА ВЫЧИСЛИТЕЛЬНОГО КОМПЛЕКСА НА БАЗЕ МИКРОПРОЦЕССОРОВ СЕМЕЙСТВА «ЭЛЬБРУС»

MANAGEMENT OF ELBRUS MICROPROCESSOR CORE POWER CONSUMTPION FROM OPERATING SYSTEM

Описываются результаты исследования, в процессе которого был определен набор стандартных состояний энергосбережения, выполнено их моделирование и определены методы управления переходом между ними. Исследование проводилось на базе специально разработанного драйвера, обеспечивающего управление состояниями энергосбережения из архитектурно-независимой части ядра Linux. Модельный эксперимент показал, что динамическое управление энергосбережением, основанное на анализе загруженности вычислительного комплекса, позволяет без потери производительности сэкономить до 20% рассеиваемой мощности.

Article reports on the results of the research of power consumption management on Elbrus microprocessor. During research a set of idle states and performance states were implemented on ASIC in purpose of modeling the hardware support of ACPI power management on Elbrus. Authors used architecture dependent implementation of linux driver, that makes possible to use decision taking algorithms from Linux kernel, that are responsible for transitions from one energy-saving state to another. According to experimental results, proposed hardware support allows to save up to 20% of power when managed by dynamic decision taking algorithms from Linux kernel.

Ключевые слова: ядро Linux, микропроцессоры «Эльбрус», управление энергопотреблением.

Keywords: Linux kernel, Elbrus, ACPI, cpuidle, cpufreq.

Введение

Одной из наиболее значимых проблем, стоящих в настоящее время перед производителями и пользователями вычислительных средств, является управление энергопотреблением. Так, плата за энергопотребление мощных серверов, входящих в состав высокопроизводительных вычислительных комплексов (ВК), стала основной статьей затрат на их обслуживание, а в категории мобильных и встраиваемых систем эффективное управление энергопотреблением позволяет улучшить одну из главных характеристик – максимальное время функционирования в автономном режиме (от батареи). В компании ЗАО «МЦСТ» исследования и разработки, связанные с этой проблемой, начались в 2011 г. в связи с проектированием процессоров для мобильных систем.

Общепринятый подход к снижению уровня энергопотребления сформулирован в стандарте ACPI [1], который в ряде распространенных архитектур поддерживается на аппаратном уровне. Согласно этому стандарту для ВК определяется несколько наборов состояния энергосбережения. Наиболее общим является набор {S0;Sn}, характеризующий энергосберегающие состояния ВК в целом с учетом процессоров, памяти, шин, периферийных устройств. Любое состояние Si из этого набора определяется тремя значениями (Ci, Pi, Di), каждое из которых принадлежит одному из трех наборов состояний, соответственно {C0;Cn}, {P0;Pn} и {D0;Dn}. Здесь {C0;Cn} – набор состояний сна для процессорного ядра, {P0;Pn} – набор активных состояний с различной частотой для процессорного ядра, {D0;Dn} – набор энергосберегающих состояний шин, памяти, контроллеров ввода вывода и периферийных устройств, расположенных на материнской плате. В соответствии со стандартом переходы из одного состояния в другое инициируются операционной системой.

В данной работе, построенной на основе проведенных ранее исследований по части поддержки стандарта ACPI в архитектуре Эльбрус [2, 3], описывается моделирование ап-

паратной поддержки наборов состояний {C0;Cn}, {P0;Pn}. Набор состояний {D0;Dn} опускается, т.к. в качестве основных потребителей энергии рассматриваются процессорные ядра. Эксперимент проведен на базе прототипа ВК, собранного из ПЛИС фирмы Altera для верификации и отладки новых разработок ЗАО «МЦСТ». Был спроектирован модуль Power Management Controller (PMC) в его составе, который соответственно программным и аппаратным воздействиям организует функционирование процессорных ядер в различных состояниях энергосбережения. Управление переходами между состояниями производится из ядра операционной системы через программные интерфейсы PMC.

Система работает под управлением ОС Linux-2.6.33, в ядре которой присутствует архитектурно-независимая реализация алгоритмов управления энергосбережением, разработанная силами Linux сообщества [4, 5, 6]. Для проведения исследования был создан драйвер, соединяющий систему с интерфейсами контроллера PMC. В результате возникла возможность проверить применимость имеющихся в ядре Linux средств к управлению энергосбережением микропроцессоров семейства «Эльбрус».

1. Аппаратная поддержка состояний энергосбережения

Обобщенная схема экспериментальной системы, включающей два процессорных ядра, кэш L2 и контроллер PMC, представлена на рис. 1. Для управления тактовой частотой, на которой работают ядра в состояниях (P0;Pn}, контроллер PMC предоставляет набор программно-доступных регистров P-state. При управлении состояниями сна {C0;Cn} в каждом ядре используется программно-доступный регистр PWRCTRL (регистр C-state).

Процессорное ядро находится в активном режиме или в состоянии сна. В активном режиме процессорное ядро может перемещаться в одно из энергосберегающих состояний набора {P0;Pn}. В. состоянии сна на нем не выполняются вычисления, и в зависимости от номера состояния в наборе {C0;Cn} отключаются дешифрация команд или синхроимпульс. Инициатором перевода ядра из одного состояния в другое является операционная

система.

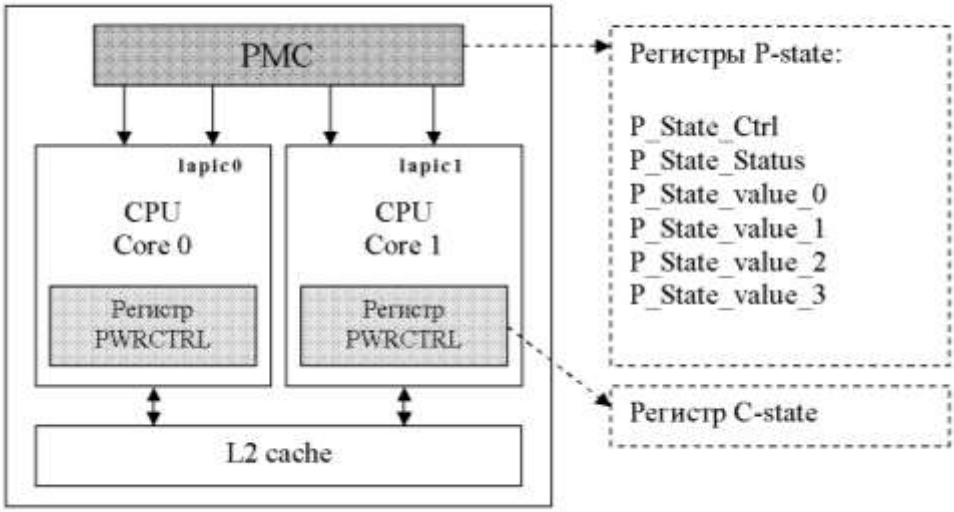

Рис. 1

Обобщенная схема системы на ПЛИС, использованной при проведении исследований

Рассмотрим аппаратную реализацию и программные интерфейсы состояний P-state, C-state более подробно.

Состояния P-state {P0;Pn}

Каждое из четырех реализованных состояний характеризуется комбинацией значений частоты синхроимпульса и напряжения питания. На этапе инициализации процедура ядра ОС записывает эти комбинации в специальные регистры P_State_value_X (X=0...3) контроллера PMC. Для рассматриваемой модели используются следующие комбинации:

P0 = {50 МГц, 0,9 В}, P1 = {40 МГц, 0,9 В}, P2 = {30 МГц, 0,9 В}, P3 = {20 МГц, 0,9 В}.

Чтобы перевести процессорное ядро из одного состояния в другое, операционная система записывает номер режима в регистр P_State_Ctrl. После того как необходимые частота и напряжение установлены, аппаратно обновляется регистр P_State_Status, показывающий текущее состояние ядра. Форматы программно-доступных интерфейсных регистров приведены в табл. 1; набор регистров является общим для обоих процессорных ядер, и частота синхроимпульса на них меняется одновременно.

Таблица 1

Форматы программно-доступных регистров контроллера PMC для управления состояниями P-state

Название регистра	Тип регистра	Значимые биты	Описание
P_State_value_X (X=0..3)	Read/Write	[15:9] – напряжение ядра; [8:4] – целая часть делителя частоты; [3:0] – дробная часть делителя частоты	Определено 4 регистра: по одному на каждое состояние P-state
P_State_Ctrl	Read/Write	[1:0] – номер состояния P-state	Управление состояниями P-state. Номера состояний: 0x0 – P0 0x1 – P1 0x2 – P2 0x3 – P3
P_State_Status	Readonly	[1:0] – номер состояния P-state	Мониторинг состояний P-state. Номера состояний: 0x0 – P0 0x1 – P1 0x2 – P2 0x3 – P3

Состояния C-state {C0;Cn}

Четыре состояния сна реализованы следующим образом:

- C0 – состояние простоя, не имеющее аппаратной поддержки, представляет собой бесконечный цикл, ожидающий прихода прерывания;
- C1 – остановка конвейера процессорного ядра;
- C2 – остановка конвейера и отключение синхроимпульса в процессорном ядре;
- C3 – самый глубокий сон, при котором останавливается конвейер, отключается синхроимпульс, и регистровый контекст ядра сохраняется в памяти.

Возврат из любого состояния сна происходит при поступлении на контроллер прерываний ядра (lapic) внешнего прерывания. Оно обрабатывается контроллером PMC, который переводит процессорное ядро в состояние простоя C0 (отметим, что при возврате из состояния C3 содержимое регистров ядра восстанавливается из памяти). Времена возврата в состояние C0 различны для разных состояний сна – так, время возврата из C1 существенно меньше времени возврата из C3.

Для управления переходами между состояниями сна, которые у двух ядер могут

быть различными, операционной системе предоставляется реализованный в каждом из них аппаратный регистр PWRCTRL. Как показано в табл. 2, программная запись определенного значения в этот регистр приводит к переходу процессорного ядра в соответствующее состояние.

Таблица 2

Значения программно-доступного регистра PWRCTRL и соответствующие им действия управления состояниями сна C-state

Название	**Тип регистра**	**Значение**	**Переход в обозначенные состояния**
PWRCTRL	Writeonly	1	C1: остановка дешифрации команд в процессорном ядре
		3	C2: остановка дешифрации команд, вытеснение значений из кэша L1, отключение синхроимпульса в процессорном ядре
		6	C3: остановка дешифрации команд, вытеснение значений из кэша L1, отключение синхроимпульса, сохранение регистрового контекста процессорного ядра в неотключаемую память, имитация отключения питания в процессорном ядре

Следует отметить, что из-за особенностей выбранной микросхемы ПЛИС отключение питания процессорного ядра с предварительным сохранением контекста программно имитируется архитектурно-зависимым драйвером. Контекст восстанавливается после прихода внешнего прерывания.

2. Программная поддержка управления состояниями энергосбережения

На уровне ОС реализуются следующие функции управления состояниями P-state и C-state:

1) анализ загруженности процессорного ядра;
2) принятие решения об оптимальных с точки зрения энергосбережения состояниях P-state и C-state в соответствии с результатом анализа загруженности;
3) перевод процессорного ядра в выбранные состояния P-state, C-state.

Первые две функции осуществляются с помощью алгоритмов принятия решений,

реализованных в архитектурно-независимой части ядра Linux [4, 5]. Третья функция выполняется архитектурно-зависимым «драйвером PMC, разработанным с учетом формата программно доступных регистров PMC и PWRCTRL.

Драйвер PMC

Драйвер PMC реализован как часть ядра Linux, выполняющая следующие функции:

- инициализация регистров P_State_value_X значениями частоты и напряжения при включении вычислительного комплекса;
- предоставление интерфейсного вызова target() для переключения между состояниями P-state в алгоритмах принятия решений;
- предоставление интерфейсного вызова enter() для переключения между состояниями C-state в алгоритмах управления состояниями сна;
- сохранение регистрового контекста при переводе процессора в состояние C3.

Фактически драйвер PMC является связующим звеном между аппаратурой, поддерживающей состояния энергосбережения, и универсальными алгоритмами принятия решений, реализованными в архитектурно-независимой части ядра. Он взаимодействует с регистрами PMC и PWRCTRL, используя виртуальные адреса специализированного пространства оперативной памяти. При инициализации системы драйвер PMC устанавливает в регистрах PMC и PWRCTRL значения, соответствующие состояниям максимальной частоты процессорного ядра (P0) и отсутствия каких-либо отключений при простое (C0). В процессе работы системы драйвер не является инициатором изменения состояний энергосбережения, решения о котором принимаются приведенными ниже алгоритмами ядра ОС.

Алгоритм выбора состояния из диапазона P-state

Для перевода процессорного ядра с одной частоты на другую универсальный алгоритм [5] использует интерфейсный вызов target(), предоставляемый драйвером PMC. Решение о том, какую частоту устанавливать, принимается в результате периодического просмотра (сэмплинга) текущей загруженности процессорного ядра. Если среднее значе-

ние загруженности в соседних точках сэмплинга ниже заданного предела, то принимается решение о переводе процессора в состояние с частотой, которая меньше текущей, иначе процессорное ядро переводится в состояние с максимальной частотой. Наименьший период сэмплинга ограничен временем перевода процессорного ядра с одной частоты на другую, но на практике это можно совместить с моментом работы планировщика процессов на конкретном ядре.

Для эффективного использования данного алгоритма аппаратура должна обеспечивать возможность быстрого переключения частот. Характерное время переключения между состояниями P-state, полученное путем расчета с использованием характеристик ПЛИС Altera, составляет 60 мс, хотя для серийного кристалла время переключения оценивается в 1 такт. Однако и при такой разнице в случае долговременного отсутствия задач на исследуемой модели можно наблюдать результат работы алгоритма.

Алгоритм выбора состояния из диапазона C-state

Выбор оптимального состояния сна зависит от статистики интервалов простоя и загруженности процессора. Оптимальным выбором будем считать состояние, переход в которое не влияет на среднюю производительность операционной системы при заданной загрузке вычислительного комплекса процессами пользователя.

Алгоритм выбора состояния сна, используемый в Linux, основан на анализе четырех факторов:

1) duration – предсказание времени нахождения процессорного ядра в состоянии простоя;
2) correction – коэффициент поправки предсказания, вычисленный на основе статистики прихода внешних прерываний;
3) latency – значение времени выхода из состояния сна;
4) latency_mutliplier – множитель для значения latency рассчитанный с использованием средней загруженности процессорного ядра.

Для принятия решения алгоритм в цикле рассматривает все состояния сна, начиная с C3, на каждой итерации проверяя условие:

$$((\text{latency} * \text{latency\_mutiplier}) < (\text{duration} * \text{correction})).$$

Если условие справедливо, то выполняется выход из цикла, и управление передается драйверу PMC, который переводит процессор в выбранное состояние сна.

Заданные в неравенстве величины вычисляются следующим образом. Значение duration рассчитывается исходя из интервала времени, на которое установлен таймер-источник внешних прерываний для планировщика процессов [7]. Это оптимистичное предсказание времени нахождения процессорного ядра в состоянии простоя. Для того чтобы сделать его более реальным, на основе статистики прихода внешних прерываний рассчитывается коэффициент correction как скользящее среднее от отношения фактического значения времени сна к оптимистической оценке, вычисленного при предыдущем простое. Если на предыдущем шаге простой продлился 50% от оптимистической оценки, то при вычислении скользящего среднего будет использован коэффициент 0,5. Значение времени выхода из состояния сна latency является константой, которая задается при инициализации драйвера PMC и фиксирована для каждого аппаратно-поддержанного состояния сна. Множитель latency_multiplier рассчитывается исходя из текущего состояния средней загруженности процессорного ядра load_average согласно эвристическому правилу:

$$\text{latency\_multiplier} = (\text{load\_average} * 10) + (\text{num\_iowaiters} * 5),$$

где num_iowaiters – количество процессов, ожидающих на данном процессорном ядре завершения обмена с устройством IO.

Описанный алгоритм является адаптивным и способен выбирать состояние глубокого сна при слабой нагрузке вычислительного комплекса, минимизируя количество переключений в одно из состояний сна при высокой нагрузке.

3. Экспериментальные результаты

В процессе исследования удалось добиться корректного взаимодействия описанных системных компонентов, относящихся к аппаратной и программной поддержкам управления энергосбережением. Модель, построенная на ПЛИС Altera, позволила провести ряд измерений, характеризующих эффективность энергосбережения. Измерения мощности, потребляемой процессорными ядрами, были выполнены для всех сочетаний состояний P-state и C-state кроме состояния самого глубокого сна C3. Показатели этого состояния не отличаются от замеров в варианте C2, т.к. отключение питания для ядра процессора на ПЛИС не моделировалось. Результаты измерений приведены в табл. 3.

Таблица 3

Мощность, рассеиваемая процессорными ядрами в различных состояниях энергосбережения

	P0, Вт	P1, Вт	P2, Вт	P3, Вт
C0	3,015	2,871	2,727	2,574
C1	2,772	2,727	2,628	2,502
C2	2,565	2,529	2,484	2,421

Измерения проводились с использованием датчиков Холла, встроенных в систему питания ПЛИС. Для сопоставления измеряемых значений с состояниями P-state, C-state использовались Linux sysfs интерфейсы универсальных динамических алгоритмов [4, 5].

Эксперимент показал, что реализованная аппаратная поддержка состояний энергосбережения в сочетании с программной поддержкой и использованием динамических алгоритмов принятия решения в ядре ОС Linux позволяют экономить до 20% мощности, рассеиваемой процессорными ядрами.

Выводы

Модельные эксперименты показали достаточную эффективность рассмотренных в статье методов управления энергосбережением процессоров семейства «Эльбрус». При проведении исследования был разработан драйвер, связывающий универсальные алго-

ритмы управления энергосбережением в ядре Linux с интерфейсами спроектированного авторами контроллера (Power Management Controller, PMC). С помощью этих средств для модели двухъядерного микропроцессора семейства «Эльбрус» на ПЛИС была достигнута экономия 20% рассеиваемой мощности.

Аппаратные и программные решения, полученные в результате исследования, рекомендованы к внедрению в новом процессоре, разрабатываемом ЗАО «МЦСТ» для мобильных устройств.

Литература

1. Advanced Configuration and Power Interface Specification, Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd, Toshiba Corporation, Revision 4.0a, April 5, 2010.

2. Кравцунов Е.М. Пути реализации стандарта ACPI 4.0 (Advanced Configuration and Power Interface Specification) для многопроцессорных вычислительных комплексов на базе процессоров «Эльбрус-E2S». – Материалы конференции молодых ученых и специалистов «Информационные технологии в автоматизированных системах управления специального назначения», ФГУП НИИАА, 6-7 сентября 2011 г.

3. Кравцунов Е.М., Семенихин С.В. Управление энергопотреблением системы на кристалле «Эльбрус-2С+» в состоянии простоя процессорного ядра. – «Вопросы радиоэлектроники», сер. ЭВТ, 2013, вып. 3, с. 143-157.

4. Venkatesh Pallipadi, Shaohua Li, Adam Belay. «cpuidle – Do nothing, efficiently …», Proceedings of the Linux Symposium 2007, Volume Two, p. 119.

5. Venkatesh Pallipadi, Alexey Starikovskiy. The Ondemand Governor: Past, Present, and Future. Proceedings of the Linux Symposium 2006, p. 657.

6. Len Brown, Anil Keshavamurthy, David Shaohua Li, Robert Moore, Venkatesh Pallipadi, Luming Yu. ACPI in Linux: Architecture, Advances, and Challenges. Proceedings of the

Linux Symposium 2005, p. 51.

7. Suresh Siddha, Venkatesh Pallipadi, Arjan Van De Ven. Getting Maximum Mileage out of Tickless. Proceedings of the Linux Symposium 2007, Volume Two, p. 201.