«Средства управления энергопотреблением микропроцессора в операционной системе Эльбрус»

Кравцунов Е.М.
kravtsunov_e@mcst.ru

26/03/2015

Agenda

- Постановка задачи
- Возможные пути решения APM, ACPI
- Стандарт ACPI, определение компонент
- Защищаемые положения
- Путь реализации
- Текущее состояние
- Публикации по теме работы

Постановка задачи

Управление энергопотреблением ВК Эльбрус из операционной системы:

- Использование микропроцессоров Эльбрус во встраиваемых системах (время автономной работы, прохождение климатических испытаний в составе бортовых систем)
- Улучшение потребительских характеристик высокопроизводительных многопроцессорных ВК Эльбрус засчет снижения энергопотребления в состоянии простоя

Постановка задачи

- Исследование способов управления энергопотреблением ВК Эльбрус из операционной системы
- Формирование предложений по аппаратной поддержке управления энергопотреблением в микропроцессорах Эльбрус, микросхеме КПИ
- Разработка программной поддержки управления энергопотреблением из ОС
- Экспериментальные исследования на макете и прототипе, моделирующих аппаратную поддержку управления энергопотреблением
- Экспериментальные исследования на опытном образце, использующем кремниевые образцы микропроцессора Процессор-2 (Эльбрус) и КПИ-2

Возможные пути решения

- APM (Advanced Power Management)

  1992-1996 Intel, Microsoft

- ACPI (Advanced Configuration & Power Interface)

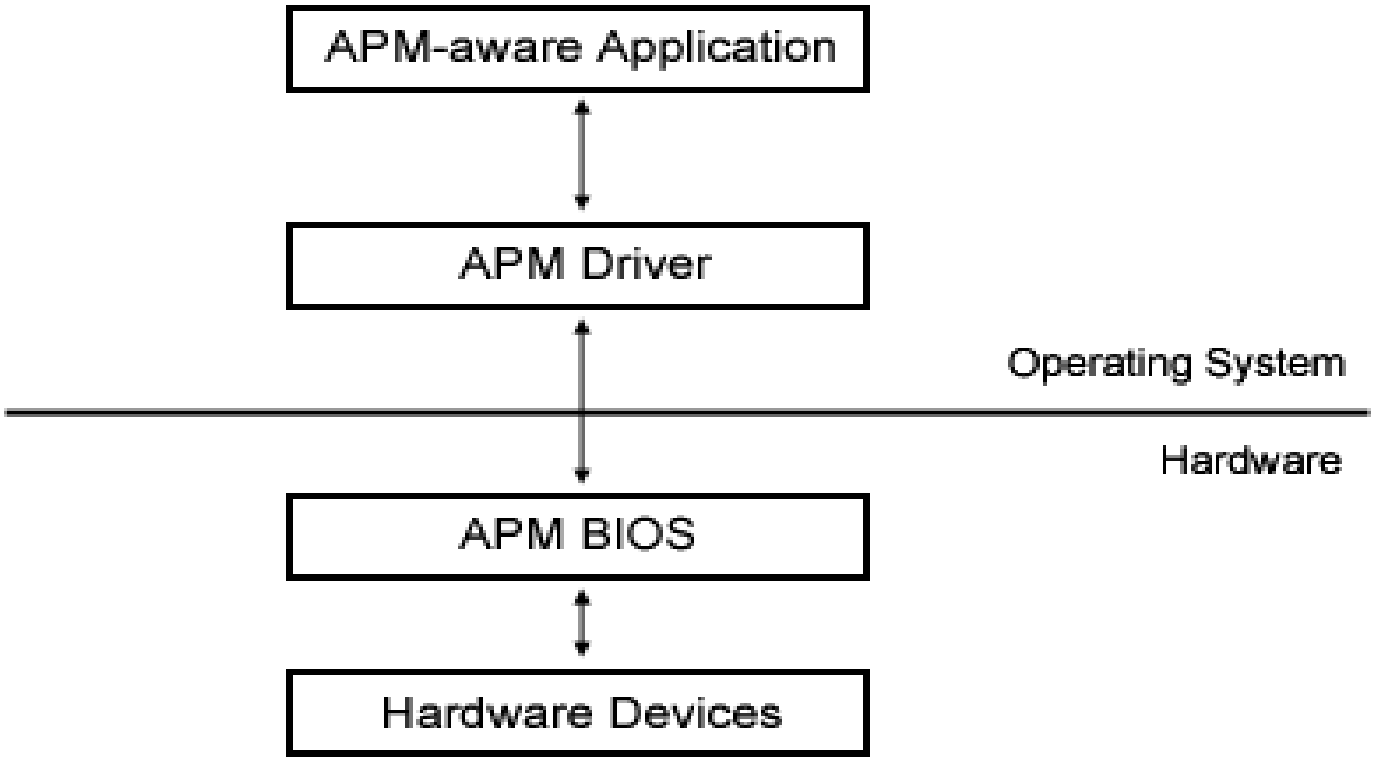
  1996-2013 HP, Intel, Microsoft, Phoenix Technologies, Toshiba

  2010: ACPI rev. 4A

  2013: ACPI rev. 5A

Возможные пути решения

- APM (Advanced Power Management)



Возможные пути решения

- APM (Advanced Power Management)

  1992-1996 Intel, Microsoft

- ACPI (Advanced Configuration & Power Interface)

  1996-2013 HP, Intel, Microsoft, Phoenix Technologies, Toshiba

  2010: ACPI rev. 4A

  2013: ACPI rev. 5A

Стандарт ACPI, наборы состояний

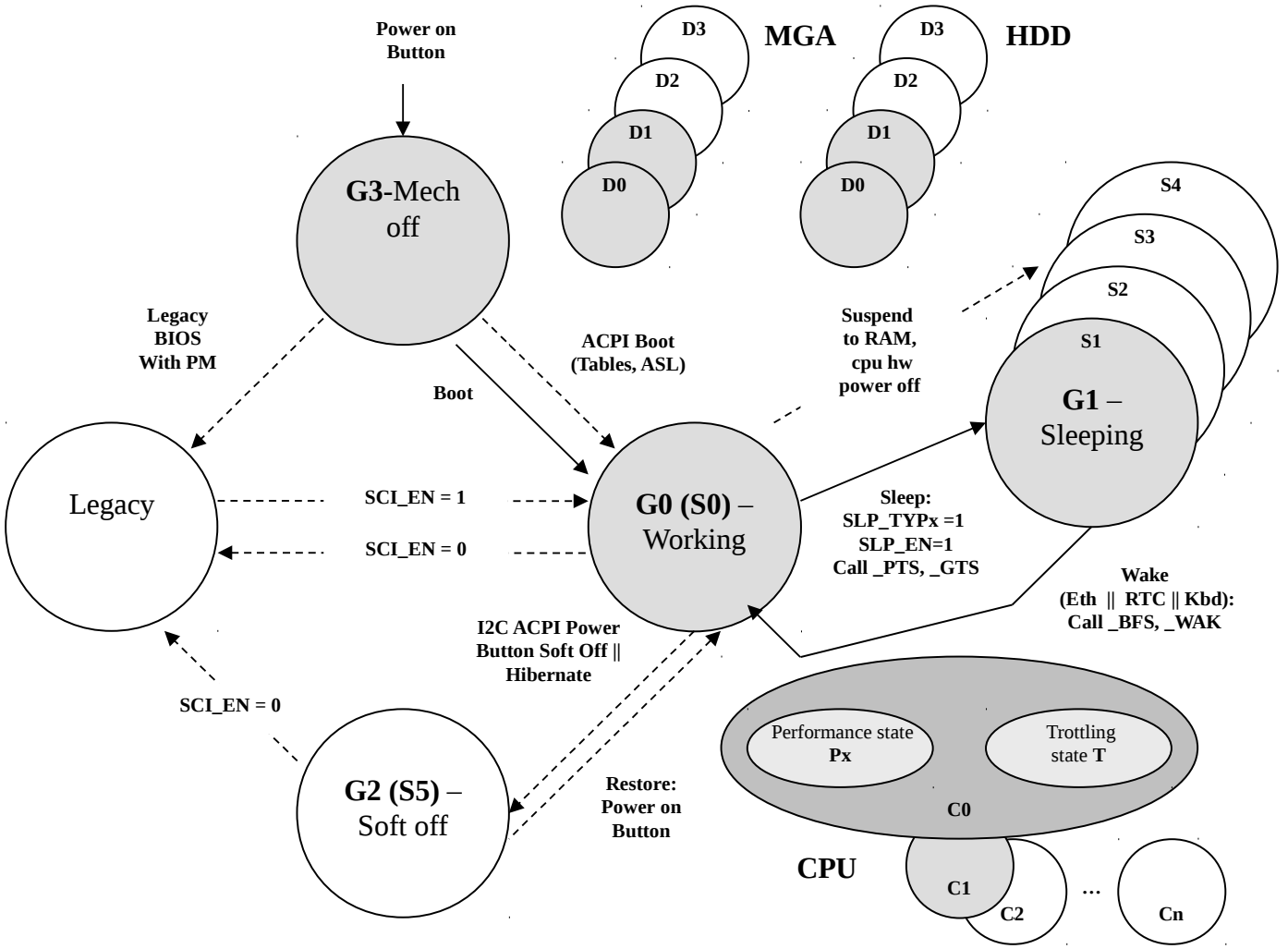
- Глобальные состояния ВК: G0, G1, G2, G3, Legacy
- Глобальные состояния сна: {S0, Sn}, где n >= 1
- Состояния P-state процессора: набор активных состояний с разными частотами синхроимпульса {P0, Pn}, T
- Состояния C-state процессора: набор состояний сна процессора {C0, Cn}
- Состояния периферийных устройств {D0, Dn}

Примеры активных состояний: Legacy | G0,

где **G0 = {S0 Pi C0 D0}, 0 <= i <= n**

Примеры состяний сна: **G1 = {S1, Ci, Dj}, 0 < i < n, 0 < j < n**

Стандарт ACPI, наборы состояний



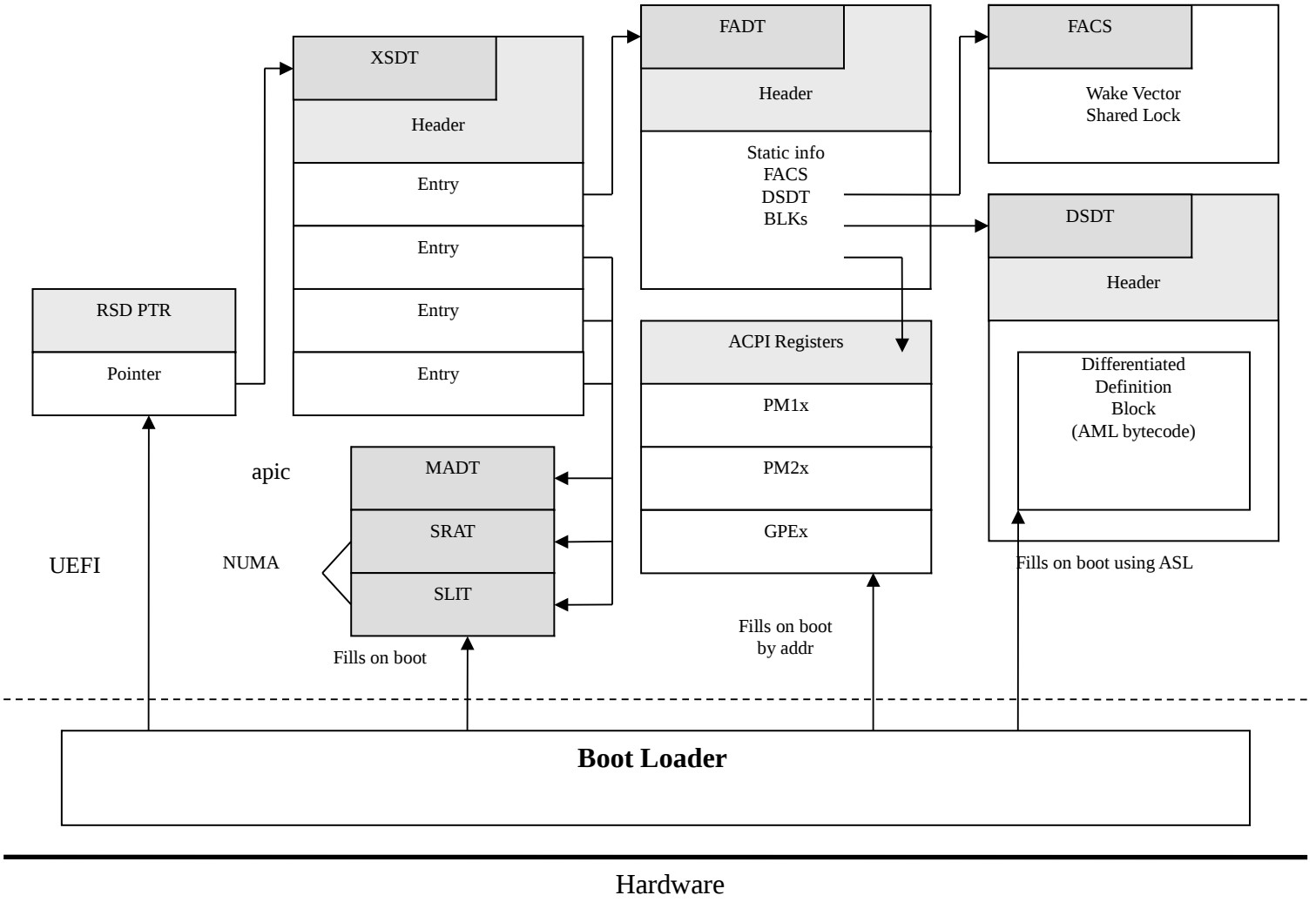
Стандарт ACPI, фукциональность

- **P-States:** Управление частотой синхроимпульса
- **C-States:** Частичное отключение CPU и устройств в состоянии простоя ВК
- **Fan:** Управление активным охлаждением (частота вращения вентилятора)
- **T:** Управление пассивным охлаждением (полное отключение синхроимпульса)
- **AC:** Отслеживание состояния источника питания
- **Battery:** Отслеживание уровня заряда батареи
- **SCI:** Информирование ОС о событии энергопотребления с использованием прерывания System Control Interrupt
- **Wakeup:** Пробуждение системы из состояния сна при появлении полезной нагрузки
- **Power button:** Кнопка перевода ВК в спящий режим

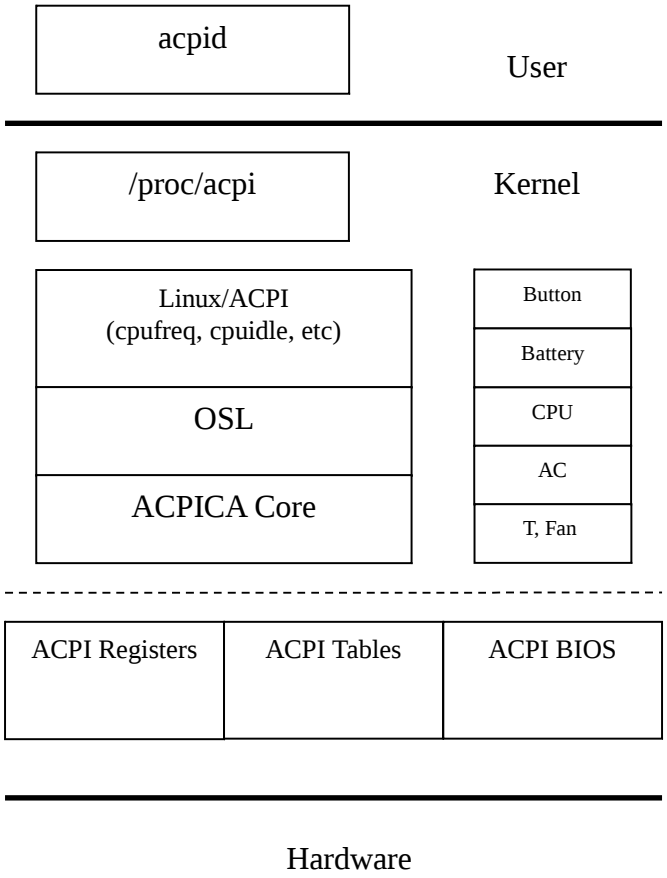
Стандарт ACPI, варианты реализации

- **ACPI Fixed** — упрощенный вариант реализации, не использующий байткода, набор событий фиксирован. Аппаратный интерфейс фиксирован
- **ACPI Full** — полный вариант реализации. Производитель микропроцессора и всего ВК определяет события энергопотребления в формате описанного в стандарте байткода. Байткод формируется загрузчиком (бут) и передается ядру ОС при инициализации. Требуется поддержка виртуальной адресации в буте, стандарта UEFI. ACPI Fixed — часть ACPI Full.

Стандарт ACPI, байткод и таблицы



Стандарт ACPI, поддержка ACPI в Linux



ОС userspace:

acpid — сервис пользователя, взаимодействующий с ядром ОС через /proc/acpi

ОС kernel:

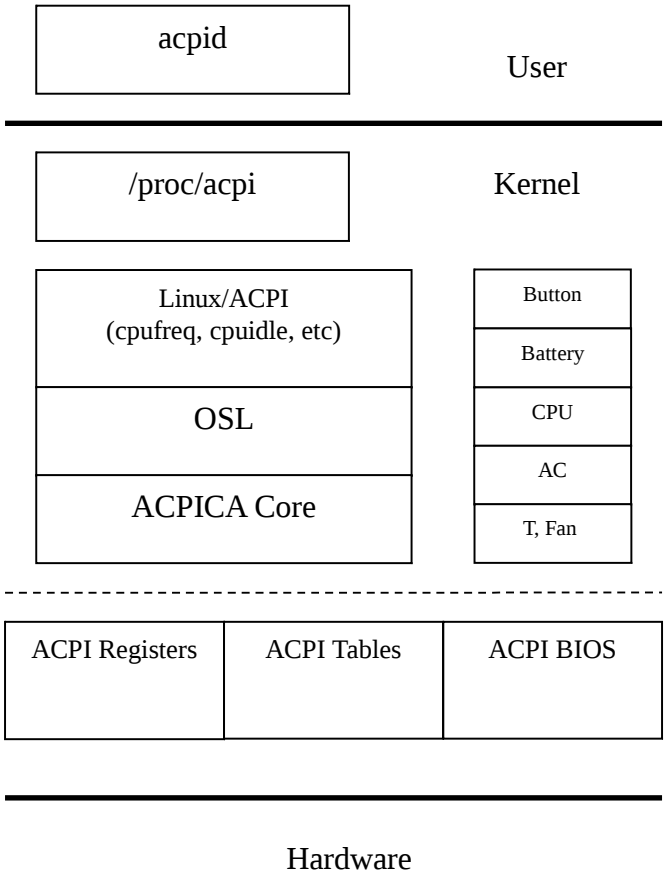
Linux/ACPI — реализации драйверов и алгоритмов принятия решений о переходе между состояниями

ACPICA Core — архитектурно и ОС-независимый компонент, реализующий алгоритмы управления состояниями

OSL — уровень стыковки с ОС

Драйверы устройств ACPI: Button, Battery, CPU, AC, Fan & T (throttling)

Стандарт ACPI, поддержка ACPI в Linux



Boot:

ACPI BIOS — виртуальная адресация, UEFI, генерация байткода. Boottime и Runtime сервисы для ядра ОС

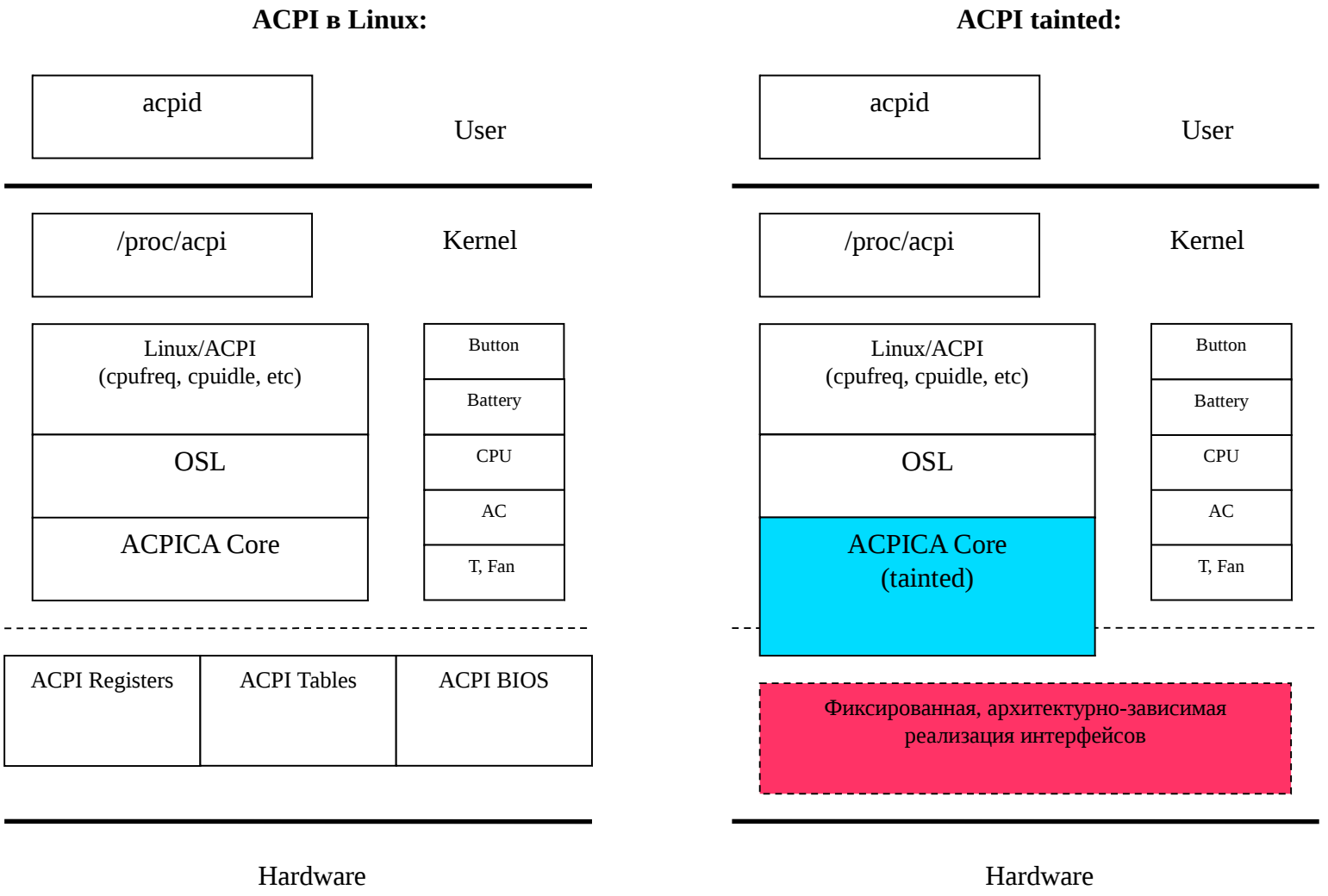
ACPI Tables — таблицы ACPI содержащие байткод

Hardware:

ACPI Registers — регистры ACPI Fixed, реализующие основную фукциональность

Стандарт ACPI, реализация tainted

Boot для Эльбрус не поддерживает и не будет поддерживать таблицы ACPI.



Защищаемые положения

- Выполнено исследование эффективности энергосбережения путем использования имеющихся в микропроцессорах Эльбрус-2с+ механизмов динамического включения/отключения синхроимпульса и дешифрации команд.
- Предложена реализация стандарта ACPI без таблиц, с использованием расширенного набора фиксированных аппаратных интерфейсов, представленных в в виде контроллера PMC (Power Management Controller) «Процессор-2» и интерфейсах микросхемы «Процессор-8».
- Реализован набор архитектурно-зависимых драйверов для управления состояниями энергопотребления из ОС. Драйверы реализованы для микропроцессоров «Эльбрус-2с+», «Процессор-2» и микросхемы «Процессор-8».
- Выполнено экспериментальное исследование эффективности управления энергопотреблением на модели ВК с поддержкой состояний P-state, C-state.

Путь реализации

1. Исследование возможностей энергосбережения на Эльбрус-2с+ (без внедрения новых аппаратных механизмов).

2. Предложения по расширенному набору фиксированных аппаратных интерфейсов. Контроллеры PMC, SPMC.

3. Создание драйвера устройства PMC. Моделирование и испытание ПО на прототипе ВК с поддержкой состояний P-state, C-state (ОКР «Изумруд»).

4. Создание драйвера устройства SPMC (КПИ-2, ОКР «Процессор-8») с поддержкой прерываний SCI (System Control Interrupts).

5. Испытания решения «Процессор-2» + «Процессор-8».

6. Внедрение наработок в НИР «Замещение» (РПКБ).

Путь реализации: исследование на Эльбрус-2с+

Защищаемое положение:

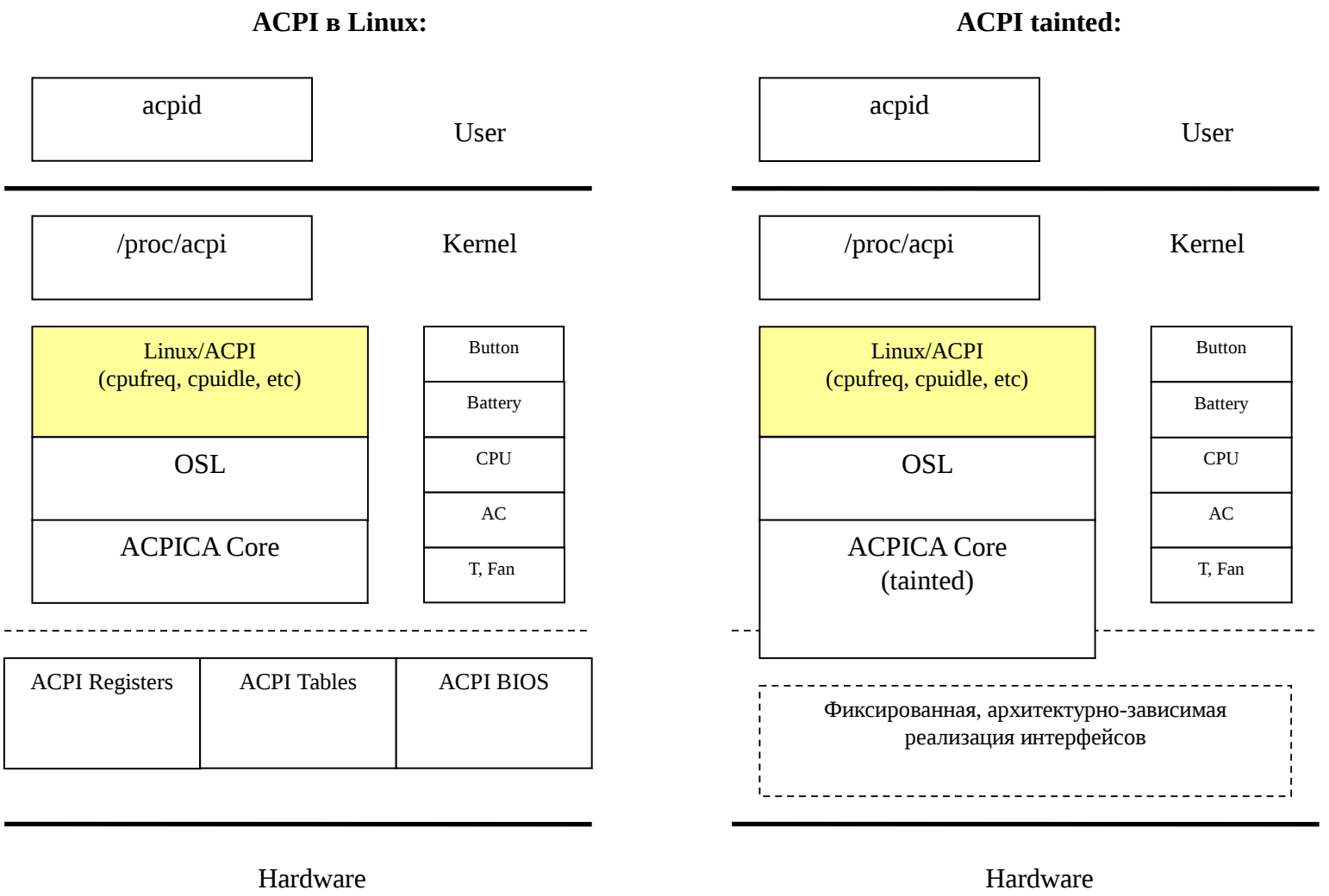
Выполнено исследование эффективности энергосбережения путем использования имеющихся в микропроцессорах Эльбрус-2с+ механизмов динамического включения/отключения синхроимпульса и дешифрации команд.

Путь реализации: исследование на Эльбрус-2с+

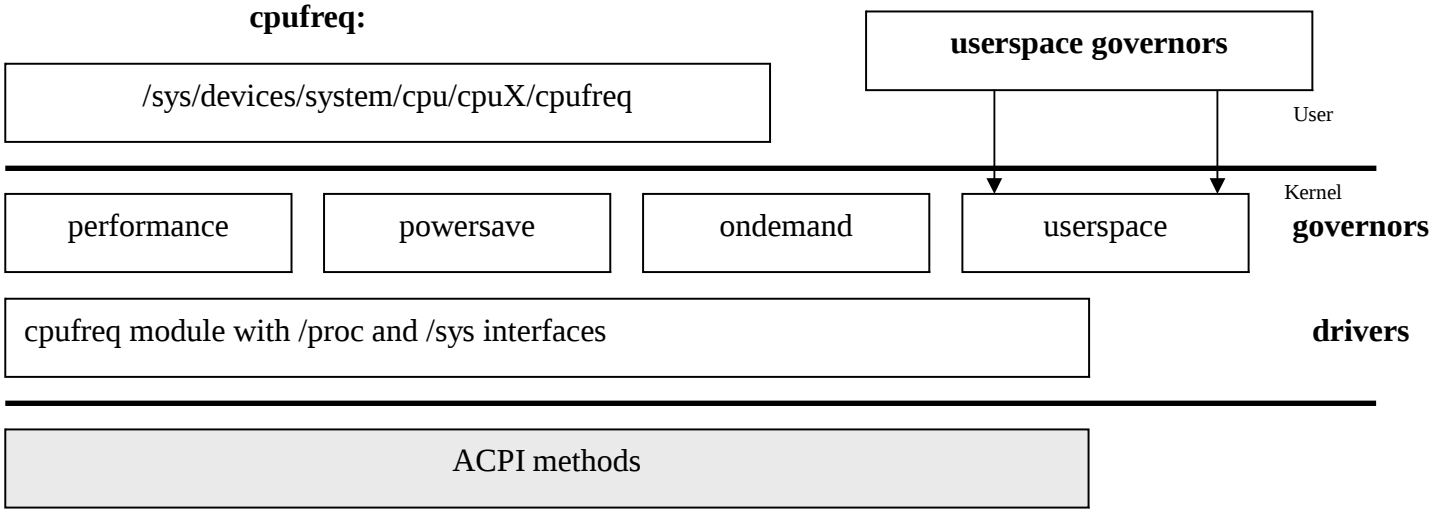
Свойства Эльбрус-2с+, позволившие исследовать эффективность энергосбережения на e2k при использовании состояний сна (C-states) микропроцессора:

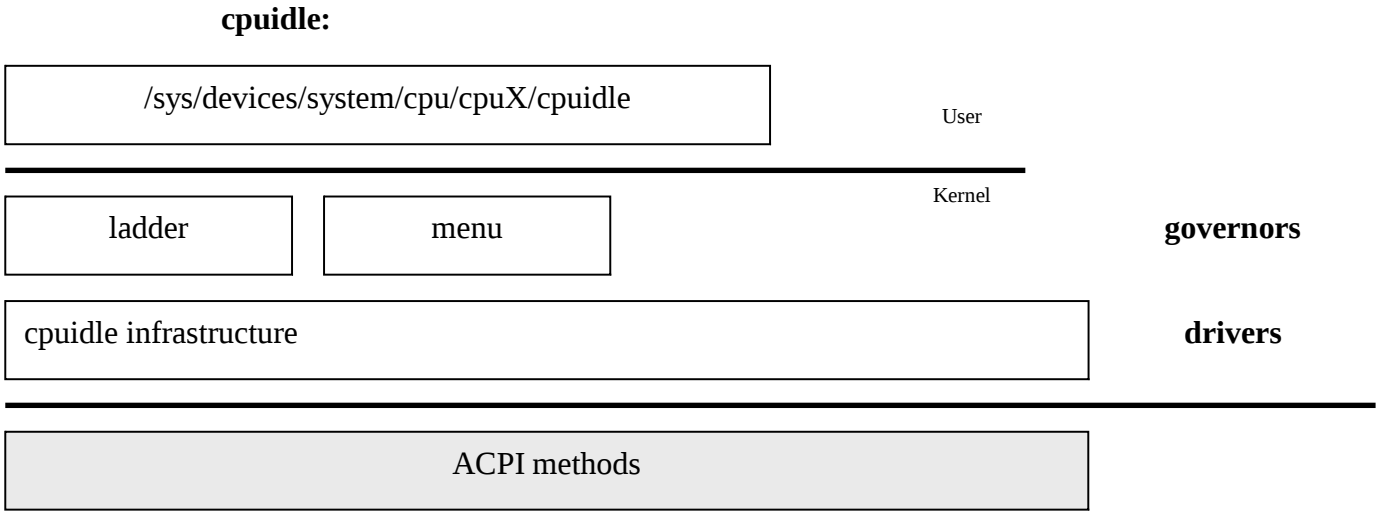
- Возможность отключения дешифрации — команда **wait trap.** Позволяет смоделировать набор состояний **C-states {C0, C1}**
- Возможность отключения синхроимпульса на одном ядре. Позволяет изучить состояние **C2** и оценить энергопотребление дерева clock'ов в микропроцессоре Эльбрус.
- Наличие таймеров с поддержкой двух режимов работы **periodic** и **oneshot** и возможностью переключения режимов в процессе работы. Позволяет исследовать эффективность имеющихся в Linux алгоритмов принятия решений о переводе микропроцессора в состояние сна при простое.

Путь реализации: исследование на Эльбрус-2с+

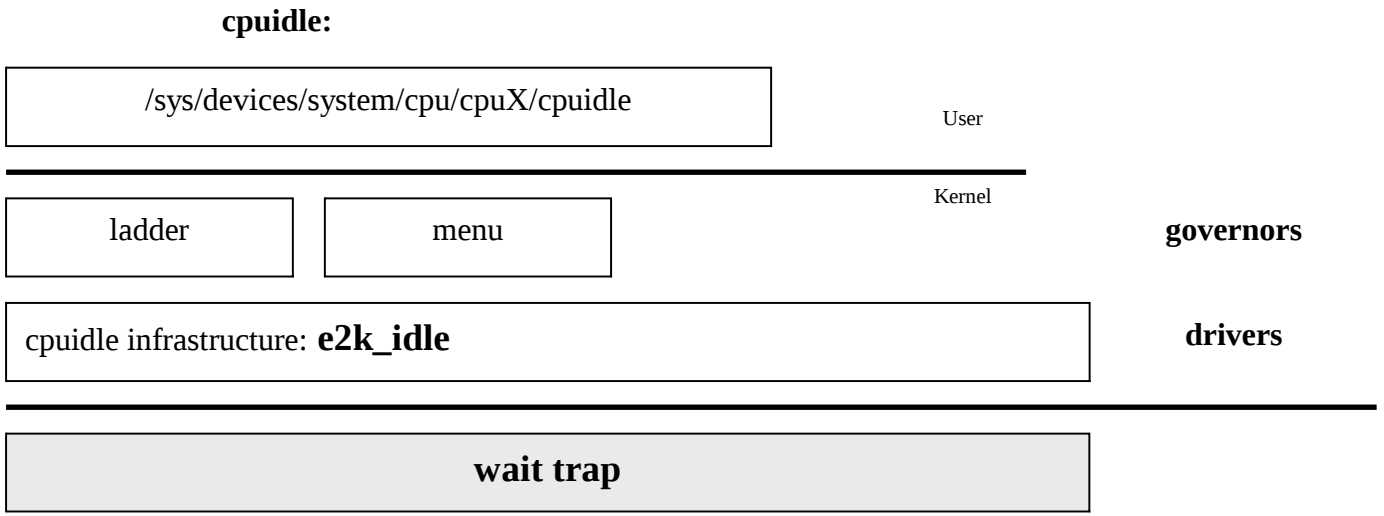


Путь реализации: исследование на Эльбрус-2с+





Путь реализации: исследование на Эльбрус-2с+



Governor — алгоритм принятия решения о переходе в определенное состояние сна. В Linux реализованы 2 алгоритма: ladder и menu.

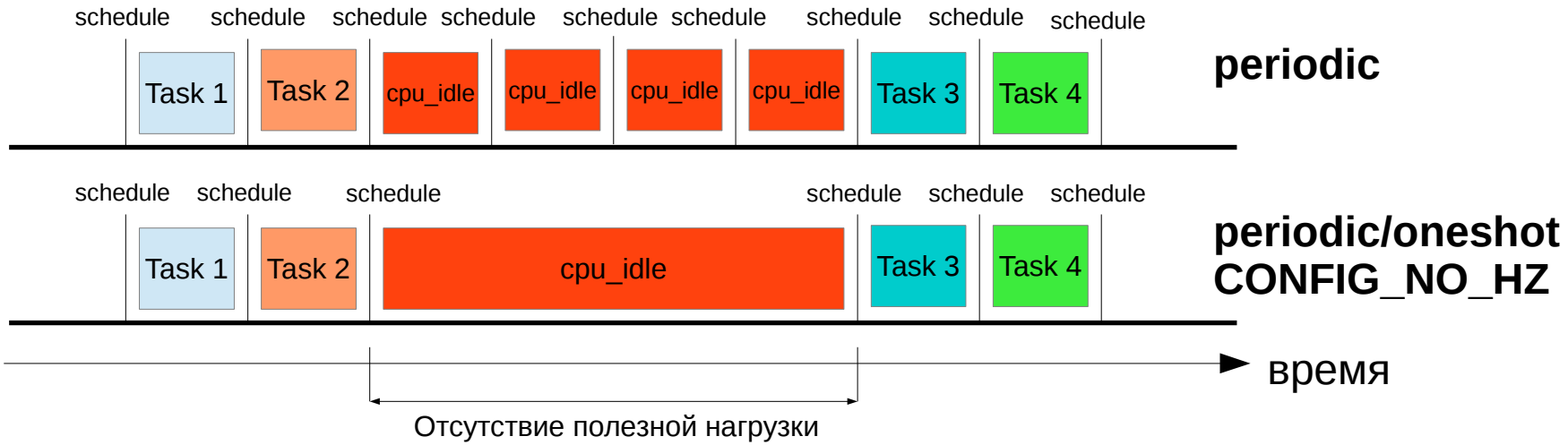
Драйвер — пассивен. Предоставляет callback-функции для governor.

Драйвер **e2k_idle**: **{C0, C1}**

C0 – простой цикл ожидания с вызовом cpu_relax на каждой итерации цикла
C1 – отключение конвейера процессорного ядра в ожидании прерывания
***C2** — отключение синхроимпульса. В динамическом варианте на Эльбрус-2с+ реализовать нельзя, но можно использовать в эксперименте по измерению тока.*

Путь реализации: исследование на Эльбрус-2с+

Динамические прерывания от таймера (CONFIG_NO_HZ):



	periodic	periodic/oneshot
прерываний/с	**101**	**15**

Продолжительность непрерываемых временных интервалов в состоянии простоя увеличивается в 7 раз

arch/e2k/kernel/process.c:

```
void cpu_idle(void)
{
                    ……….
  /* Tell NOHZ code that we are in the idle loop */
  tick_nohz_stop_sched_tick(1);

   while (!need_resched())  {
          ……..

   }
   /* Disable NOHZ mode */
   tick_nohz_restart_sched_tick();
    ……..
}
```

Путь реализации: исследование на Эльбрус-2с+

Временные параметры governor menu и governor ladder

	При отсутствии нагрузки ВК				При наличии нагрузки ВК			
	Ladder		**Menu**		**Ladder**		**Menu**	
	T_{C0}, %	**T_{C1}, %**	**T_{C0}, %**	**T_{C1}, %**	**T_{C0}, %**	**T_{C1}, %**	**T_{C0}, %**	**T_{C1}, %**
Cpu0	16,63	83,37	3,07	96,93	3,28	96,72	73,95	26,05
Cpu1	2,57	97,43	2,58	97,42	0,38	99,62	76,5	23,5
Average	**9,6**	**90,4**	**2,83**	**97,17**	**1,83**	**98,17**	**75,23**	**24,77**

Для governor'ов ladder и menu оценивались относительные интервалы времени нахождения вычислительных ядер процессора «Эльбрус-2С+» в состояниях С0 и С1 в двух вариантах: простой ВК в течение 30 с (отсутствие нагрузки); работа ВК, загруженного счетной задачей raytrace из набора splash-2.

T_{CO} – время, проведенное в состоянии простоя, в котором энергия не экономится (цикл ожидания новой активности);

T_{C1} – время, проведенное в состоянии сна, устанавливаемом средствами ОС.

Путь реализации: исследование на Эльбрус-2с+

Значения тока в трех вариантах нагрузки с учетом отключения синхроимпульса

	2 ядра			**1 ядро**		
	Нагрузка	Простой **C0**	Состояние сна **C1**	Нагрузка	Простой **C0**	Состояние сна **C1**
Ток, А	**2,27**	**2,17**	**2,11**	**1,77**	**1,7**	**1,67**

Эффективность энергосбережения в состояниях C-state в расчете на 1 ядро cpu:

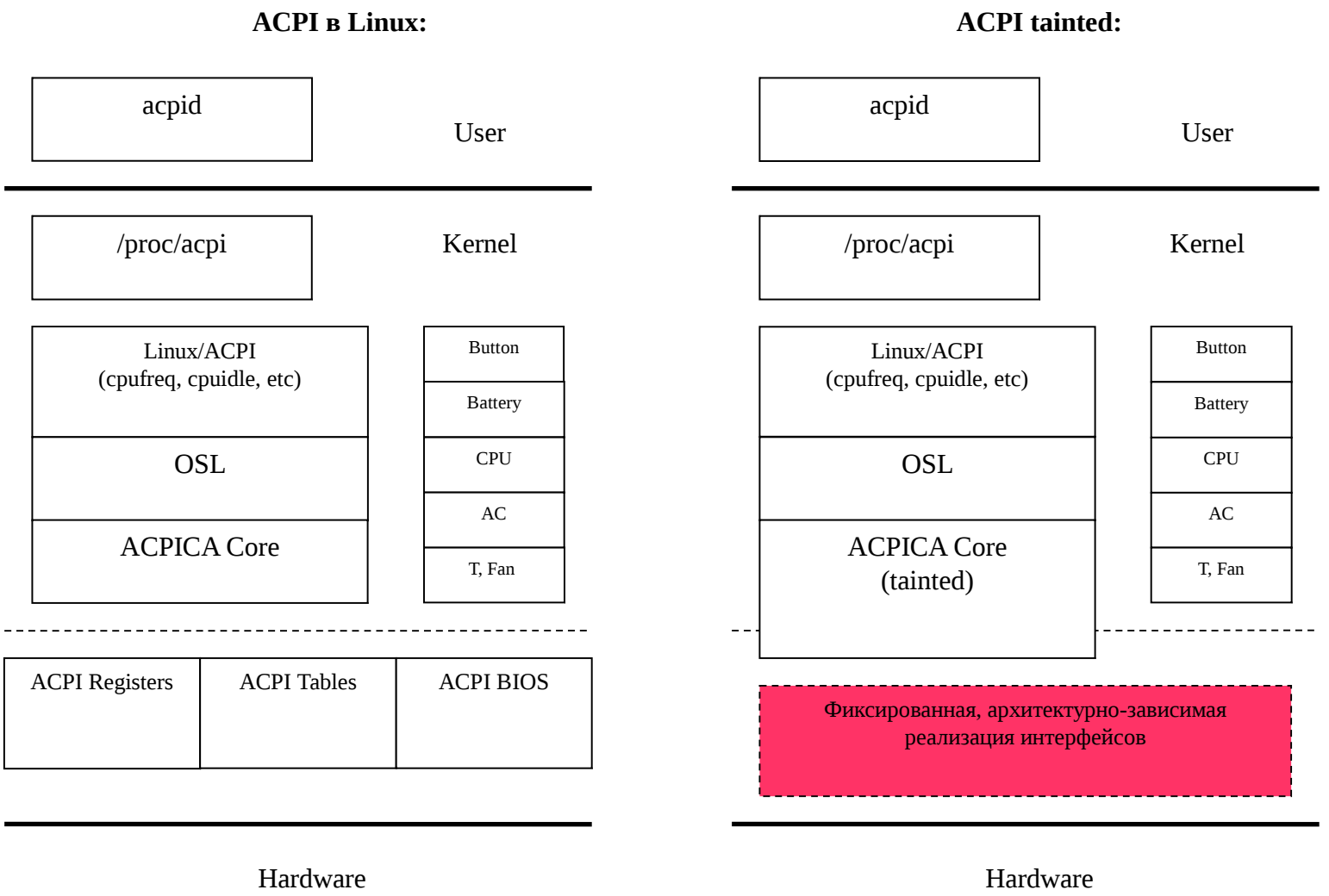
Состояние	Экономия (**%** от потребления в нагруженном состоянии)
C0	**4,1**
C1	**5,6**
C2	***26,3***

Путь реализации: аппаратная поддержка

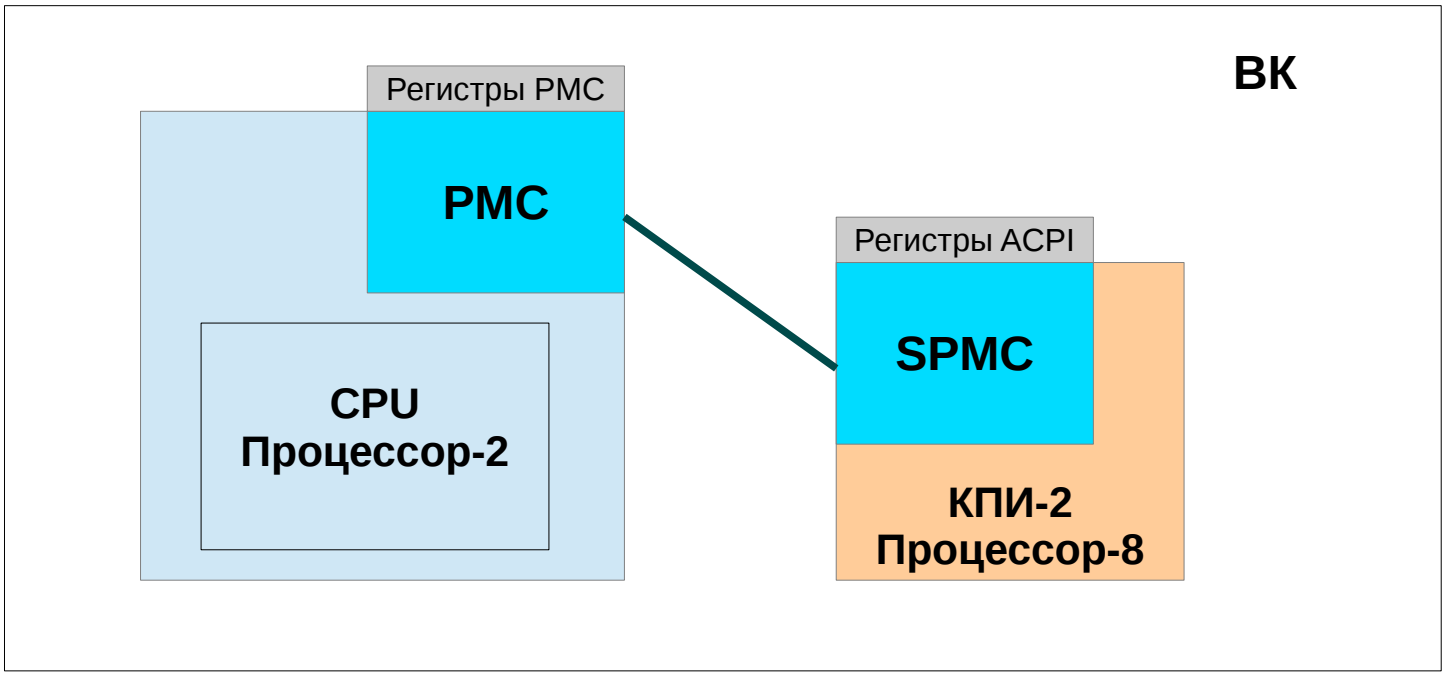
Защищаемое положение:

Предложена реализация стандарта ACPI без таблиц, с использованием расширенного набора фиксированных аппаратных интерфейсов, представленных в виде контроллера PMC (Power Management Controller) «Процессор-2» и интерфейсах микросхемы «Процессор-8».

Путь реализации: аппаратная поддержка



Путь реализации: PMC, SPMC



PMC (Power Management Controller):

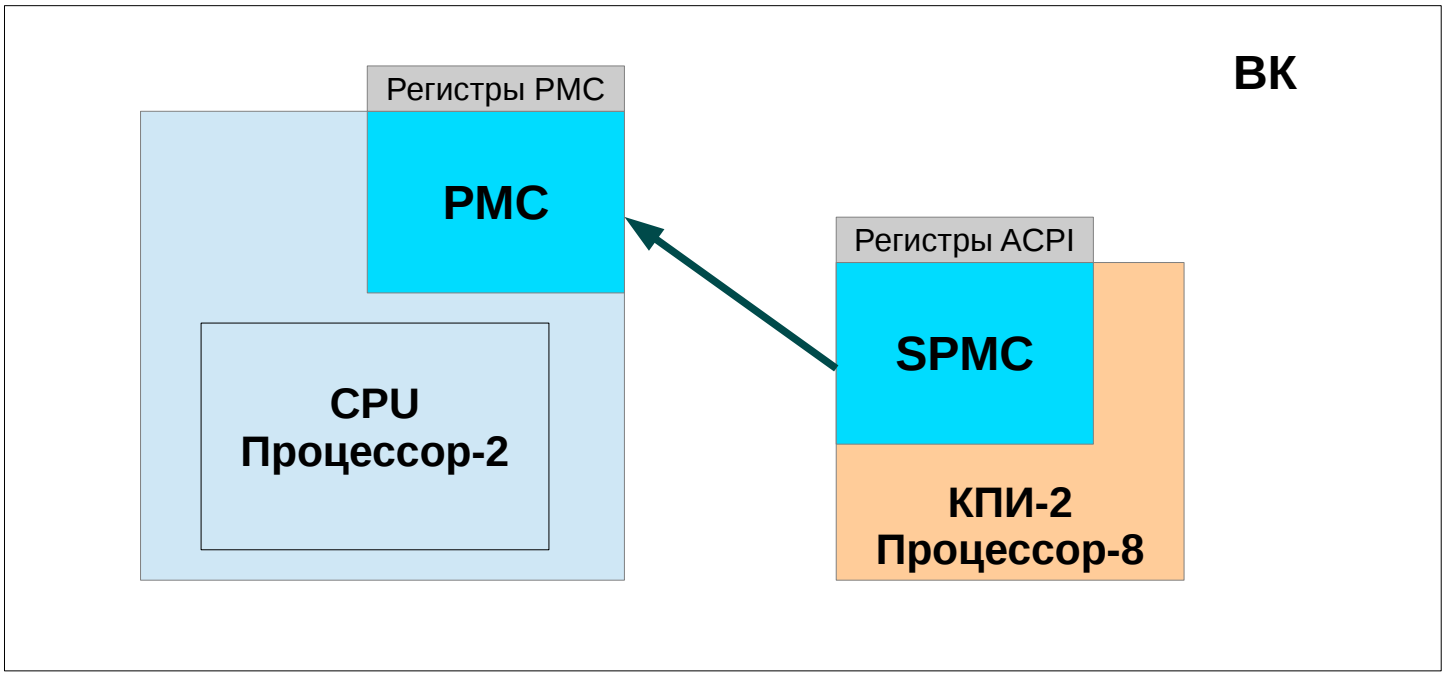
1) P-states {P0....P3}
2) C-states {C1...C4}
3) События температуры

SPMC (System Power Management Controller):

Регистры ACPI:

1) SCI прерывания
2) Кнопка Power
3) События от источника питания AC
4) События от батареи
5) Таймер простоя (PM Timer)
6) События wakeup

Путь реализации: PMC

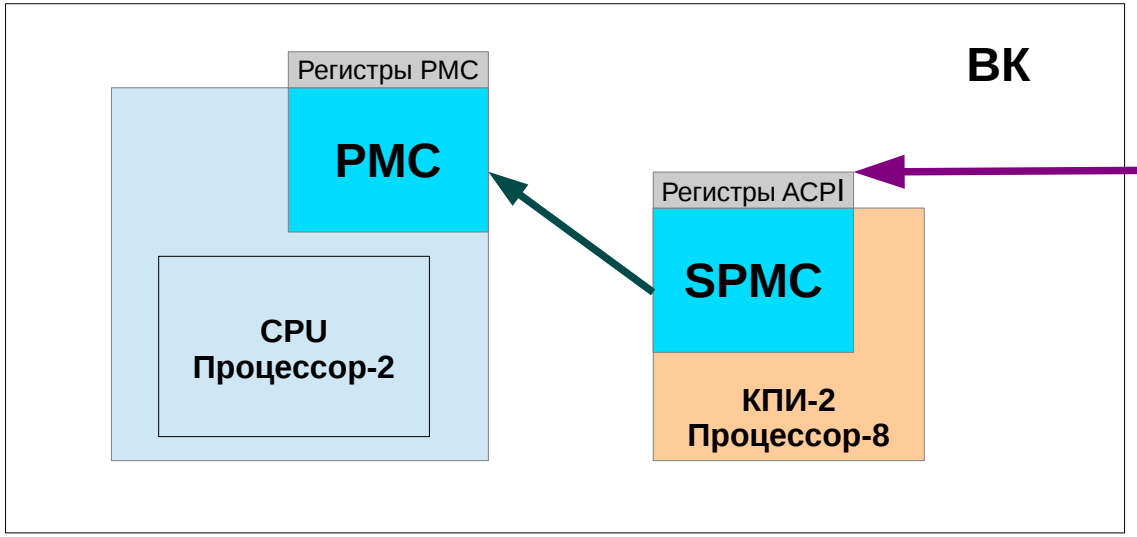


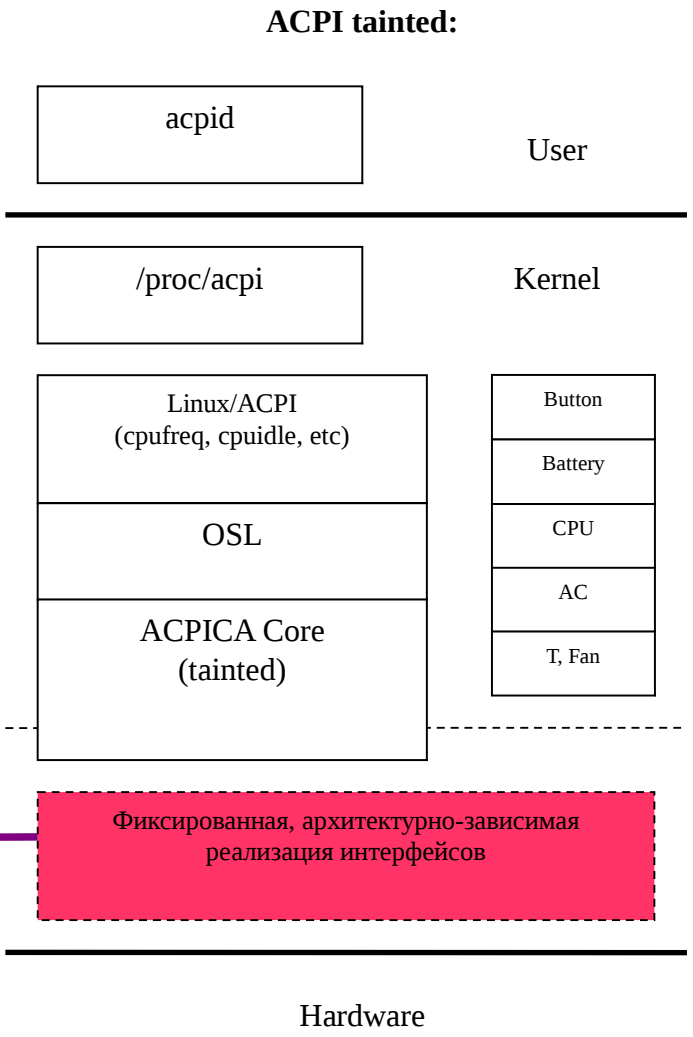
PMC реализует наборы состояний **P-states**, **C-states**.
C-states содержит 3 состояния сна, различающиеся по глубине и возможностям снижения энергопотребления:

C1 – остановка конвейера процессорного ядра;
C2 – остановка конвейера и отключение синхроимпульса на процессорном ядре;
C3 – состояние C2 и сохранение регистрового контекста процессорного ядра в область retention.

Путь реализации: PMC, SPMC

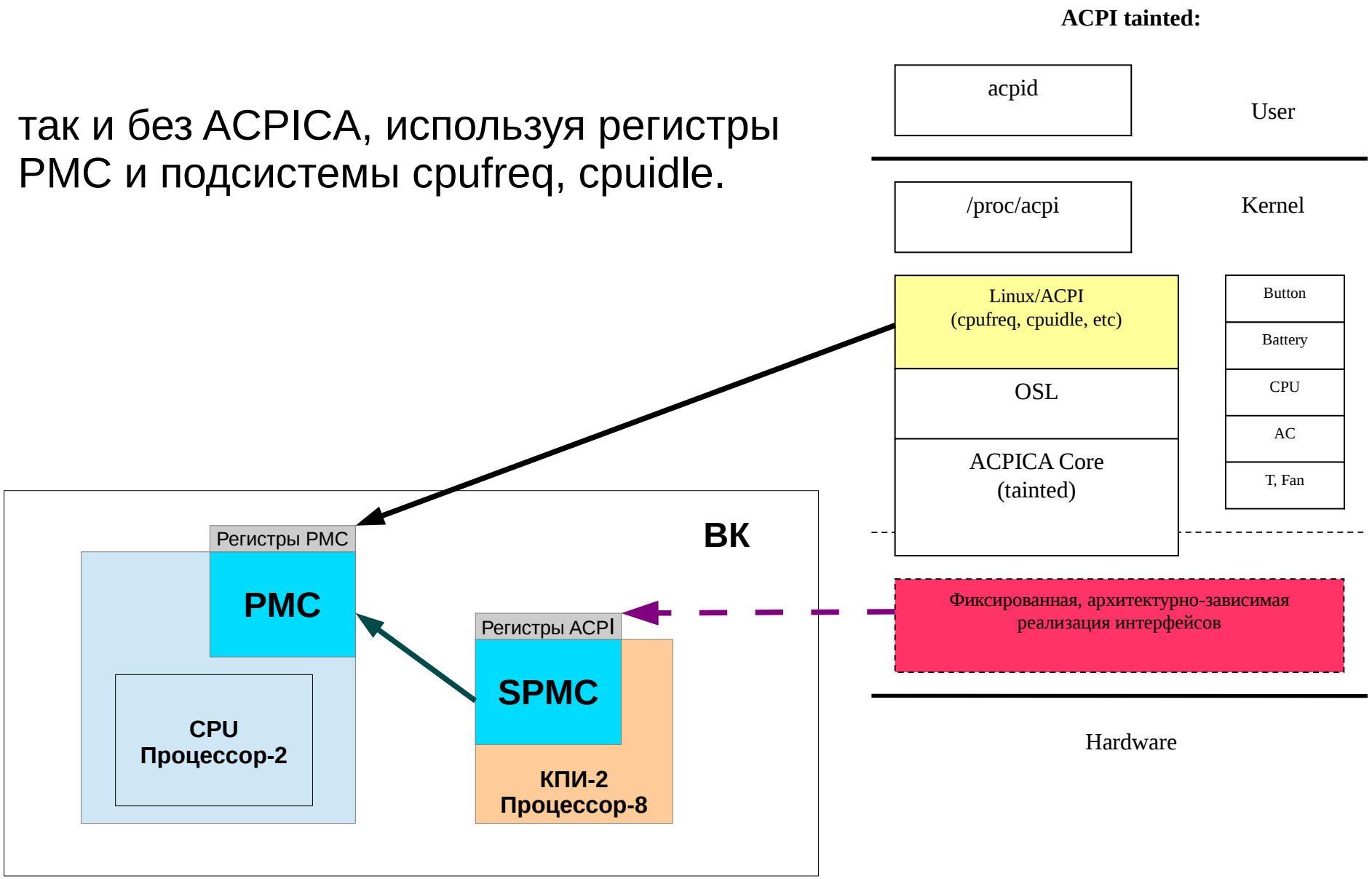
Два набора программно доступных регистров (регистры PMC и регистры ACPI в SPMC) позволяют выполнять эксперимент как с использованием ACPICA, реализованным в Linux





Путь реализации: PMC, SPMC

так и без ACPICA, используя регистры PMC и подсистемы cpufreq, cpuidle.



Путь реализации: регистры ACPI Fixed в SPMC

Регистр PM1_STS

Номер бита	Тип бита	Название
0	Read Only	**TMR_STS**
1-3	Reserved	
4	Status	**BM_STS**
5	Status	**GBL_STS**
6-7	Reserved	
8	Status	**PWRBTN_STS**
9	Status	**SLPBTN_STS**
10	Status	**RTC_STS**
11	Ignore	
12-14	Reserved	
15	Status	**WAK_STS**

Регистр PM1_EN

Номер бита	Тип бита	Название
0	Write Only	**TMR_EN**
1-3	Reserved	
4	Reserved	
5	Write Only	**GBL_EN**
6-7	Reserved	
8	Write Only	**PWRBTN_EN**
9	Write Only	**SLPBTN_EN**
10	Write Only	**RTC_EN**
11	Reserved	
12-14	Reserved	
15	Reserved	

Путь реализации: регистры ACPI Fixed в SPMC

Регистр PM1_STS

Номер бита	Тип бита	Название
0	Read Only	**TMR_STS**
1	Read Only	**AC_PWR_STATE**
2	Status	**AC_PWR_STS**
3	Read Only	**BATLOW_STATE**
4	Status	**BATLOW_STS**
5	Status	**GBL_STS**
6-7	Reserved	
8	Status	**PWRBTN_STS**
9	Reserved	***SLPBTN_STS***
10	Reserved	***RTC_STS***
11	Ignore	
12-14	Reserved	
15	Status	**WAK_STS**

Регистр PM1_EN

Номер бита	Тип бита	Название
0	Write Only	**TMR_EN**
1	Read Write	**TMR_32**
2	Write Only	**AC_PWR_EN**
3	Reserved	
4	Write Only	**BATLOW_EN**
5	Write Only	**GBL_EN**
6-7	Reserved	
8	Write Only	**PWRBTN_EN**
9	Reserved	***SLPBTN_EN***
10	Reserved	***RTC_EN***
11	Reserved	
12-14	Reserved	
15	Reserved	

Путь реализации: регистры ACPI Fixed в SPMC

Регистр PM1_CNT

Номер бита	Тип бита	Название
0	Read Only	**SCI_EN**
1	*RW*	***BM_RLD***
2	*Write Only*	***GBL_RLS***
3-8	Reserved	
9	Ignore	
10-12	RW	**SLP_TYPx**
13	Write Only	**SLP_EN**
14-15	Reserved	

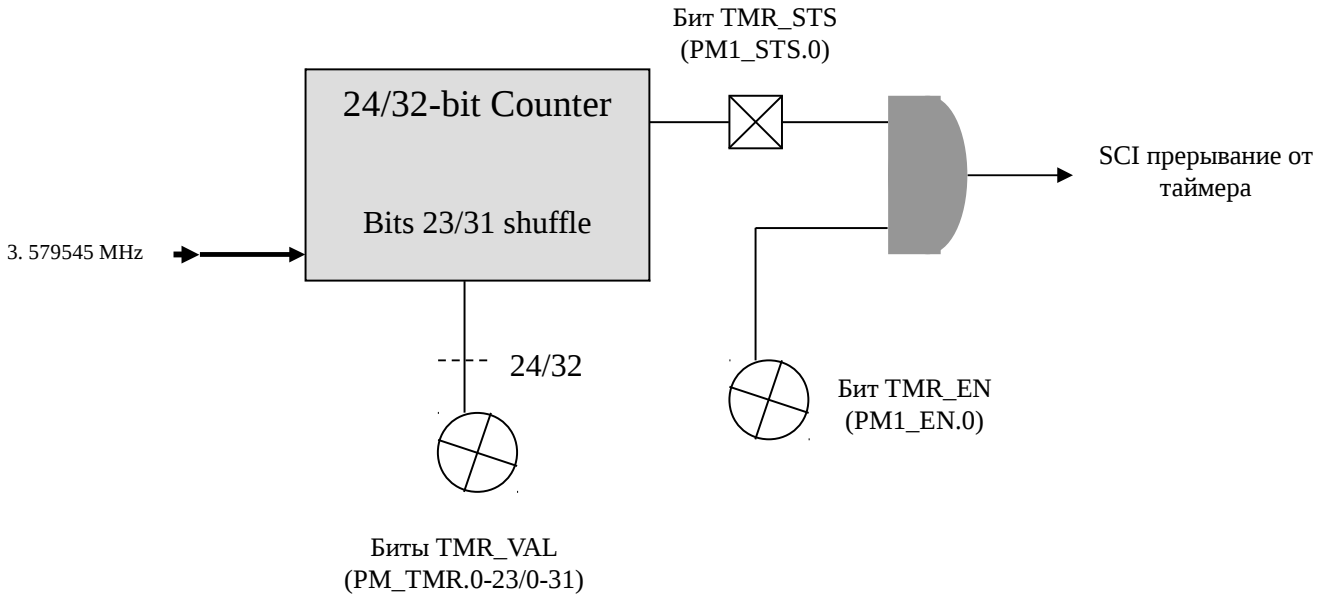
Путь реализации: регистры ACPI Fixed в SPMC

Регистр PM1_CNT

Номер бита	Тип бита	Название
0	Read Only	**SCI_EN**
1	Reserved	***BM_RLD***
2	Reserved	***GBL_RLS***
3-8	Reserved	
9	Ignore	
10-12	RW	**SLP_TYPx**
13	Write Only	**SLP_EN**
14-15	Reserved	

Путь реализации: регистры ACPI Fixed в SPMC

Таймер простоя PM Timer



Регистр PM_TMR

Номер бита	Тип бита	Название
0-23	Read Only	**TMR_VAL**
24-31	Read Only	**E_TMR_VAL**

Путь реализации: регистры ACPI Fixed в SPMC

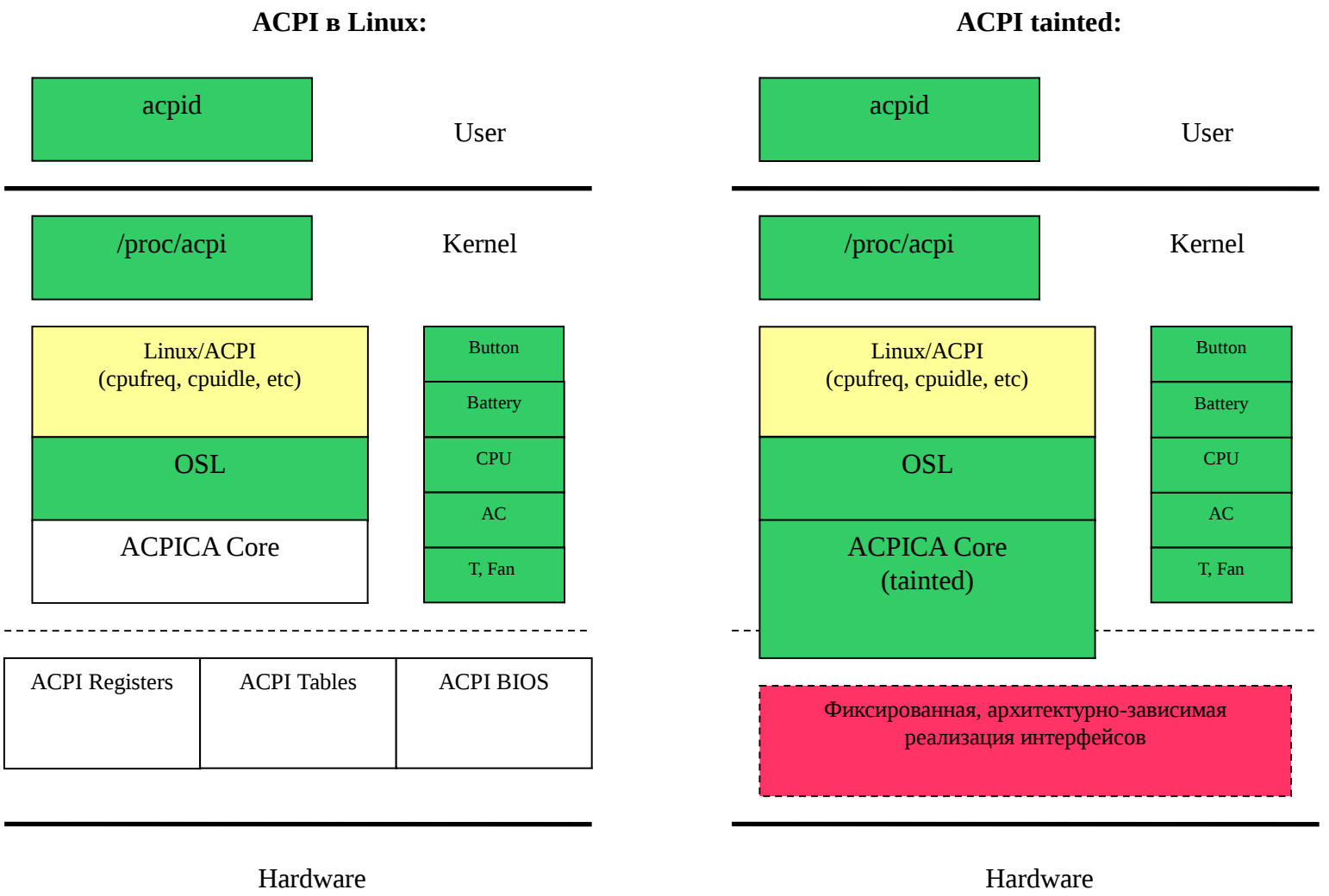
Условие формирования SCI прерывания

```
SCI = SCI_EN &  (
            (TMR_EN & TMR_STS)  |
                (AC_PWR_EN &  AC_PWR_STS) |
                (BATLOW_EN &  BATLOW_STS) |
                (G0 & PWRBTN_STS & PWRBTN_EN) |

                (G1 & WAK_STS)

            );
```

Путь реализации: ACPI tainted



Путь реализации: программная поддержка

Защищаемое положение:

Реализован набор архитектурно-зависимых драйверов для управления состояниями энергопотребления из ОС. Драйверы реализованы для микропроцессоров «Эльбрус-2с+», «Процессор-2» и микросхемы «Процессор-8».

Путь реализации: программная поддержка

Архитектурно зависимые драйверы

Компонент	Путь в дереве исходных текстов ядра ОС Эльбрус (linux-2.6.33)	Описание
e2k_idle	**arch/e2k/kernel/cpuidle.c**	Исследование на Эльбрус-2с+
PMC	**arch/l/kernel/pmc.c**	Моделирование состояний P-states, C-states и прораммная поддержка «Процессор-2»
SPMC	**arch/l/kernel/acpi/spmc.c**	Поддержка SCI прерываний и обработка событий контроллера SPMC

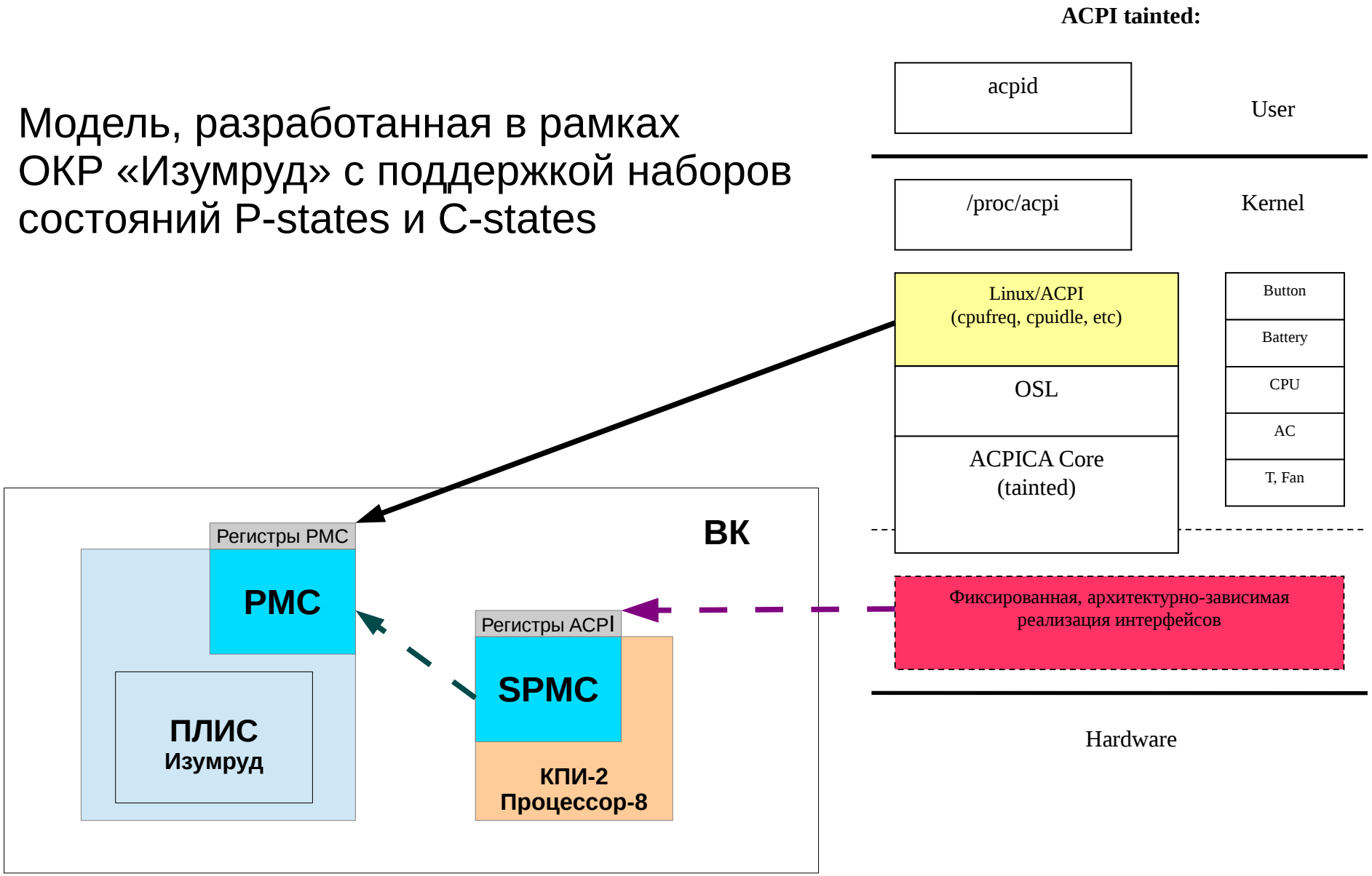
Путь реализации: модель (ОКР Изумруд)

Защищаемое положение:

Выполнено экспериментальное исследование эффективности управления энергопотреблением на модели ВК с поддержкой состояний P-state, C-state.

Путь реализации: моделирование

Модель, разработанная в рамках ОКР «Изумруд» с поддержкой наборов состояний P-states и C-states



Путь реализации: модель (ОКР Изумруд)

Результаты измерений

Измерения проводились с использованием датчиков Холла, встроенных в систему питания ПЛИС.

Мощность, рассеиваемая ядрами cpu 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**

Экономия относительно {P0; C0} составляет **20%**

P-states: {50 МГц, 40 МГц, 30 МГц, 20 МГц}

C-states:
C0 — активный простой
C1 — отключение дешифрации команд в ожидании прерывания
C2 — отключение дешифрации команд и синхроимпульса в ожидании прерывания

Состояние самого глубокого сна не моделированось.

Текущее состояние

Микропроцессор	Архитектура	Возможные реакции на SCI- прерывание
Эльбрус-2С+	Эльбрус	✔ Выключение одного процессорного ядра микропроцессора путем отключения синхроимпульса
Эльбрус-4С	Эльбрус	✔ Выключение от одного до трех процессорных ядер ✔ Перевод всех ядер микропроцессора в состояние глубокого сна путем отключения синхроимпульса. Выход из состояния глубокого сна происходит при приходе внешнего прерывания.
МЦСТ R1000	Sparc V9	✔ Выключение от одного до трех процессорных ядер микропроцессора путем отключения синхроимпульса
Процессор-2	Эльбрус	✔ Снижение частоты синхроимпульса до минимально возможной. ✔ Перевод ядра микропроцессора в состояние глубокого сна путем отключения синхроимпульса. Выход из состояния глубокого сна происходит при приходе внешнего прерывания. ✔ Комбинация вариантов 1 и 2.

Публикации по теме работы

- Кравцунов Е.М. «Пути реализации стандарта ACPI 4.0 для многопроцессорных вычислительных комплексов на базе процессора Эльбрус-2S», Конференция молодых ученых и специалистов, посвященная 55-летию со дня образования НИИ автоматической аппаратуры им.В.С.Семенихина, **2011**
- Е.М. Кравцунов, С.В. Семенихин «Управление энергопотреблением СНК “Эльбрус-2С+” в состоянии простоя процессорного ядра», Вопросы радиоэлектроники серия «Электронная вычислительная техника», выпуск 3, **2013**
- Волин В.С., Кравцунов Е.М., Семенихин С.В., Фельдман В.М., Черепанов В.М. «Управление энергопотреблением процессорных ядер из операционной системы для прототипа ВК на базе микропроцессоров семейства “Эльбрус”», Вопросы радиоэлектроники, серия «Электронная вычислительная техника», Выпуск 3, **2014**-04-09
- Кравцунов Е.М., Михайлов М.С., Семенихин С.В. «Использование прерываний системного контроля SCI для управления энергопотреблением микропроцессоров семейства “Эльбрус”», Вопросы радиоэлектроники, серия «Электронная вычислительная техника», Выпуск 3, **2015**

Вопросы

Backup

Backup исследование на Эльбрус-2c+. C-states: «Menu» governor

Алгоритм выбора состояния сна, реализованный в “Menu” governor, основан на анализе четырех факторов:

duration - предсказание времени нахождения процессорного ядра в состоянии простоя

correction - коэффициент поправки предсказания, вычисленный на основе статистики прихода внешних прерываний

latency – значение времени выхода из состояния сна

latency_mutliplier - множитель для значения latency рассчитанный с использованием значения средней загруженности ядра cpu.

Для принятия решения алгоритм “Menu” в цикле обходит все возможные состояния сна начиная с состояния наиболее глубокого сна. На каждой итерации цикла выполняется проверка условия **((latency * latency_mutiplier) < (duration * correction))**. Если условие выполняется, то происходит выход из цикла с сохранением номера состояния сна и упраление передается архитектурно-зависимому драйверу, который и выполняет перевод процессора в выбранное состояние сна.

Значения факторов, формирующих условие, вычисляются governor'ом следующим образом.

Значение duration вычисляется с использованием алгоритма поддержки динамических прерываний от таймера (CONFIG_NO_HZ): duration рассчитывается из текущего времени, на которое взведен таймер-источник внешних прерываний для планировщика процессов. Duration представляет собой оптимистичное предсказание времени нахождения процессорного ядра в состоянии простоя. Оценка является оптимистичной, так ка предполагает, что за время duration не будут приходить никакие внешние прерывания. Для того чтобы сделать оценку более реальной, на основе статистики прихода внешних прерываний рассчитывается коэффициент correction. Коэффициент correction рассчитывается как скользящее среднее от отношения реального значения времени сна к оптимистической оценке на 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 для governor является константой, оно определяется при инициализации архитектурно-зависимого драйвера cpuidle. Это значение является фиксированным для каждого поддержанного аппаратно состояния сна.

Множитель latency_multiplier рассчитывается в governor исходя из ткеущего состояния средней загруженности процессорного ядра load_average. Для расчета используется следующее эвристическое правило: **latency_multiplier = (load_average * 10) + (num_iowaiters * 5)**, где **num_iowaiters** – количество процессов, ожидающих на данном процессорном ядре завершения обмена с устройством IO.

Таким образом governor “Menu” использует адаптивный алгоритм, способный выбирать состояния глубокого сна при слабой загрузке вычислительного комплекса и минимизировать переводы процессорного в состояния сна при высокой загрузке.

Backup исследование на Эльбрус-2c+. C-states: функция cpu_idle

arch/e2k/kernel/process.c:

```
 ……
/*
 * Powermanagement idle function, if any..
 */
void (*pm_idle)(void);
EXPORT_SYMBOL(pm_idle);

 ……
/*
* We use this if we don't have any better
* idle routine..
*/
void default_idle(void)
{
        int cpu = raw_smp_processor_id();

        if (psr_and_upsr_irqs_disabled()) {
                local_irq_enable();
        }

        /* loop is done by the caller */
        cpu_relax();
}
EXPORT_SYMBOL(default_idle);

void __cpuinit select_default_idle_routine()
{
       pm_idle = default_idle;
}
```

```
void cpu_idle(void)
{
    int cpu = raw_smp_processor_id();
    int cpuup = 0;

    while (1) {
            while (!need_resched()) {
                    rmb(); /* check the case carefully */
                    if (cpu_is_offline(cpu)) {
                      play_dead();
                     cpuup = 1;
                     break;
                    }
                    /* Process RCU */
                    ……
                    local_irq_disable();
                    pm_idle();
            }
   .....
     }
}
```

Backup исследование на Эльбрус-2с+. C-states: выбор состояния сна и вход в него

drivers/cpuidle/cpuidle.c:

.....

```
static void cpuidle_idle_call(void)
{
    struct cpuidle_device *dev = __get_cpu_var(cpuidle_devices);
    struct cpuidle_state *target_state;
    int next_state;
    /* ask the governor for the next state */
      next_state = cpuidle_curr_governor->select(dev);
     if (need_resched()) {
        local_irq_enable();
        return;
     }
    target_state = &dev->states[next_state];
    /* enter the state and update stats */
    dev->last_state = target_state;
    dev->last_residency = target_state->enter(dev, target_state);
    if (dev->last_state)
       target_state = dev->last_state;

    target_state->time += (unsigned long long)dev->last_residency;
    target_state->usage++;

    /* give the governor an opportunity to reflect on the outcome */

    if (cpuidle_curr_governor->reflect)
        cpuidle_curr_governor->reflect(dev);
 }
```

arch/e2k/kernel/cpuidle.c:

….

```
/* Interface for entering the sleep state */
static int e2k_enter_idle(struct cpuidle_device *dev,
                                    struct cpuidle_state *state)
{
    ktime_t before, after;
    thread_info_t *thread_info = current_thread_info();
    thread_info->wtrap_jump_addr = PG_JMP;
    before = ktime_get();
    if (state == &dev->states[0]) {
       while (!need_resched()) {
       default_idle();
       }
    } else if (state == &dev->states[1]) {
       SET_WTRAP_JUMP_ADDR("jump_over_wtrap");
       local_irq_enable();
       wtrap(); /* wait trap=1 */
       JUMP_OVER_WTRAP_LABEL("jump_over_wtrap");
    }
    if (thread_info->wtrap_jump_addr == PG_JMP) {
       after = ktime_get();
       thread_info->wtrap_jump_addr = 0UL;
    } else {
       after = (ktime_t) (thread_info->time_in_idle_ns);
    }
    return ktime_to_ns(ktime_sub(after, before)) >> 10;
}
```

Appreciates

Я являюсь частью коллектива МЦСТ. И мне это очень нравится.

Спасибо всем кто сделал эту работу и ее внедрение возможными: С.В. Семенихин, В.И. Перекатов, В.С. Волин, Ф.А. Груздов, С.А. Черепанов, М.С. Михайлов, В.М. Фельдман, И.Н. Бычков, А. Федоров, разработчики ОС, компилятора, разработчики документации и ребята из наладки.

Пусть дарует вам Бог здоровья и успехов в обоих Мирах (земном и вечном).