УДК 004.051 DOI: 10.31114/2078-7707-2022-3-108-112

# Методы реализации быстрой загрузки встраиваемых вычислительных систем

Бычков И. Н., Ломако С.Г., Ломако Г.А.

ПАО ИНЭУМ, им. И. С. Брука, г. Москва, Россия, bignat@mail.ru

***Аннотация* — В этой статье рассмотрены методы сокращения до нескольких секунд времени загрузки от подачи питания до старта инициализации операционной системы. Рассмотренные методы являются универсальными, и для реализации некоторых из них нужна аппаратная поддержка. Эти методы испытаны в вычислительных комплексах на основе процессоров Эльбрус и проанализирована их эффективность.**

***Ключевые слова* — быстрая загрузка, загрузчик, быстрый старт, оптимизация времени загрузки, многопоточные вычисления, оптимизация.**

## I. Введение

На растущем рынке встраиваемых вычислительных систем быстрая загрузка вычислительных комплексов до готовности является одним важных требований в таких сферах как автоматика, авионика, медицина, потребительские устройства и так далее.

В статье рассмотрена загрузка вычислительных комплексов в момент после подачи питания. В этот момент в оперативной памяти нет операционной системы. Само по себе аппаратное обеспечение не может выполнять сложные действия, такие как, например, загрузку программы в память. Решением данной проблемы является использование специальной программы начального старта (ПНС) [1]. Данные ПНС хранятся в энергонезависимой памяти [2], откуда и начинается инициализация оборудования, после чего, выполняется инициализация операционной системы.

Проблемы быстрой загрузки ПНС рассматривают такие разработчики как Texas Instruments, разработчики операционной системы с открытым исходным кодом Android, разработчики U-Boot, coreboot, LinuxBoot.

Методы, которые реализуют все разработчики ПНС, схожи. Методы программного ускорения в основном такие:

1. Отключение вывода в консоль
2. Инициализация и установка драйверов только необходимого оборудования.
3. Использовать ПНС в конфигурации, которая позволит после SPL (сокращение от second stage bootloader) [3] напрямую начать распаковку ядра операционной системы.
4. Сжатие ядра операционной системы.

В частности разработчики ПНС U-Boot реализовали специальный режим «Falcon» [3] позволяющий после инициализации необходимого оборудования перейти напрямую к инициализации ОС и отдать дальнейшую настройку всех устройств ОС. Приведённые замеры времени загрузки [4] показали, что время загрузки ПНС в специальном режиме сократится на 2.32 секунд (рис.1, 2).

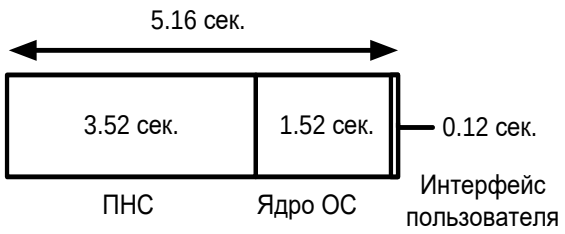


**Рис. 1. Время загрузки U-Boot в обычном режиме**

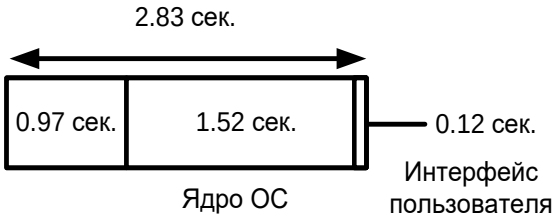


**Рис. 2. Время загрузки U-Boot в специальном режиме**

На рис. 1 и 2 приведены диаграммы, где ПНС – время работы ПНС, Ядро ОС – время работы ядра ОС, Интерфейс пользователя – сокращённое время загрузки интерфейса пользователя до старта первого приложения.

В статье рассмотрены методы и как совокупность решений для ускорения загрузки встроенной ПНС. Реализация этих методов выполнена в ПНС вычислительного модуля на основе процессора E2K Эльбрус [5].

## II. Особенности выбора и подключения ПЗУ быстрой загрузки

Постоянное запоминающее устройство (ПЗУ) [2] - хранилище всех необходимых данных для инициализации оборудования за исключением ПНС. Исходя из этого, можно говорить о том, что обмен информацией между ПНС и необходимыми ему данными для инициализации аппаратуры будет уменьшать время загрузки. Использование FRAM [2]

в качестве ПЗУ это относительно эффективное решение. Принципиальная схема данного решения представлена на рис. 3.

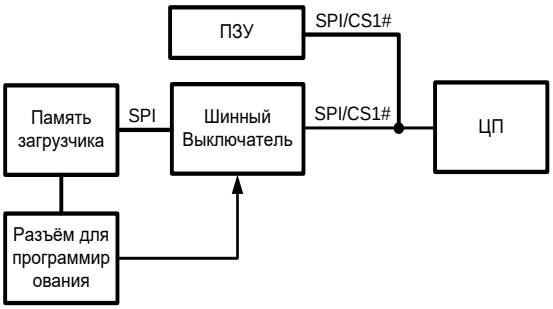


**Рис. 3. Блок схема с FRAM в качестве ПЗУ**

Так, как аппаратным интерфейсом к FRAM является SPI [6] то использование этого компонента аналогично использованию NVRAM. Так же FRAM обеспечивает более высокую скорость доступа и позволяет совершить большее количество циклов чтения записи ($\sim 10^{15}$) чем NVRAM ($\sim 10^{6}$). Достоинством для NVRAM же будет являться больший объём памяти устройства. Стоит отметить, что пропускная способность SPI интерфейса ниже, чем у FRAM в связи с этим можно говорить, что SPI ограничивает на аппаратном уровне скорость загрузки. И даже не смотря на недостатки пропускной способности SPI, FRAM показывается результат на ~0.4 секунды превосходящий NVRAM. Всё вышеперечисленное даёт основания полагать, что использование FRAM с аппаратным интерфейсом с более высокой пропускной способностью даст более ощутимый выигрыш во времени загрузки.

## III. Методы ускорения загрузки

Список основных инструментов, обеспечивающих ускорение:

A. Широковещательная настройка физ. уровней и контроллеров памяти.
B. Параллельная инициализация ядер процессора в режиме многопоточности.
C. Отключение вывода в консоль.
D. Обмен сообщениями через регистры.
E. Быстрый кооперативный планировщик без приоритетов.
F. Спин-блокировка для синхронизации доступа к ресурсам.
G. Расширенная настройка параметров работы.

### *A. Широковещательная настройка физ. уровней и контроллеров памяти*

Широковещательный обмен сообщениями через регистры это метод, реализуемый на уровне архитектуры, быстрой настройки ОЗУ путем рассылки широковещательных сообщений контроллерам памяти и физическим уровням. Широковещательный режим опирается на технологию Cache-As-Ram(CAR) [7]. CAR позволяет использовать кэш процессора в качестве ОЗУ до инициализации самого ОЗУ.

Алгоритм настройки:

1. Широковещательно запрещаем работу всех каналов.
2. В режиме отладки покажем частоту памяти.
3. Запомним номер канала памяти плюс рабочая установка канала.
4. Запускаем физические уровни DDR процессора широковещательно.
5. Если нет сбоя физ. уровня, то по каждому каналу памяти локально ждем статуса завершения настройки этого физ. уровня.
6. Далее регулируем уровень опорного напряжения для стабилизации питания процессора.
7. Широковещательно настраиваем ряд регистров контроллеров DDR: TIM0/2, CFG, PERF0, CTL и пр. Тут регистры DDR приведены для примера, метод будет работать и для регистров других DDR главное правильно сконфигурировать рабочую установку для них.
8. После настройки физ. уровней широковещательно инициализируем контроллеры памяти.
9. Далее в цикле локально по каждому каналу памяти ждем завершения инициализации.
10. Если включен режим отладки, показываем состояния всех регистров нулевого физ. уровня и нулевого канала памяти.
11. Если нужно, отрабатываем широковещательно для каналов памяти режим самообновления.
12. Как итог, выдаем полученную максимальную маску каналов памяти в широковещательном режиме.

### *B. Параллельная инициализация ядер центрального процессора в режиме многопоточности*

Многопоточность – состояние системы, при котором вычисления производятся одновременно и не зависимо друг от друга. Эти вычисления могут быть связанны по смыслу или не зависимы друг от друга. В частности, ПНС E16C.V6, U-Boot, Coreboot, LinuxBoot применяют многопоточность для сокращения времени запуска ядер центрального процессора. Базовый алгоритм — последовательный. В нем ядра центрального процессора настраиваются в цикле одно за другим. В многопоточном режиме кооперативный планировщик главного ядра процессора раздает ведомым побочным ядрам задания на запуск путем рассылки программных сообщений. В свою очередь, ведомые ядра после самонастройки отправляют ему сообщения об окончании инициализации. Процесс заканчивается, когда планировщик дождался ответа от всех реципиентов.

### C. *Возможность отключение вывода в консоль*

Включение режима тишины в параметрах настройки ПНС приводит к тому, что функции TTY перестают генерировать выходные данные. Какой практический результат наличия данной опции? За счет запрета вывода в консоль удалось реально повысить скорость загрузки. Кроме того в режиме SILENCE все технологические паузы автоматом приводятся к минимуму вплоть до 0.

### D. *Обмен сообщениями через регистры*

Для ускорения загрузки ядер процессора сообщения доставляются через NBSR регистры как самый быстрый вариант или через оперативную память. При этом каждому ядру соответствует свой регистр NBSR или значение в массиве в оперативной памяти.

### E. *Спин-блокировка для синхронизации доступа к ресурсам*

Доступ к общим ресурсам (RAM, TTY, FLASH и пр.) синхронизируется через Спин-блокировку — низкоуровневый примитив синхронизации, применяемый в многопроцессорных системах для взаимного исключения исполнения на базе цикла активного ожидания. Спин-блокировка применяется, когда ожидание захвата предполагается недолгим или контекст выполнения не предполагает перехода в состояние пассивного ожидания.

## IV. Алгоритм инициализации

Проверенный на практике и приемлемый по эффективности алгоритм инициализации такой:

1) После подачи питания распаковывается ядро с необходимым код для работы с оборудованием, которое детектирует материнскую плату и запускает средства самотестирования.
2) Инициализируется южный мост.
3) Инициализируется внешние протоколы обмена данными I2C [8] и SPI.
4) Инициализируется консоль для ввода текстовых команд.
5) Конфигурируются линки.
6) Нумеруются процессоры
7) Разрешается когерентность
8) Подкачиваются данные с энергонезависимой памяти и данные содержащие неинициализированные глобальные переменные [9].
9) Формируются стеки.
10) Определяются тип и размер ОЗУ.
11) Широковещательно настраивается ОЗУ. Главное ядро выполняет маршрутизацию, создает карту регионов памяти и выделяет место для размещения кучи.
12) После настройки ОЗУ наступает очередь инициализации ядер процессора. Вызывается планировщик, который ищет ядра процессора, которые обрабатывают задачи и многопоточно развертывает их.
13) АР-ядра настраивают себя, выполняя действия схожие с теми, что делались в первом пункте.
14) После развертывания ядер процессора строятся деревья устройств подключённых по шине ввода-вывода PCI [10], устанавливаются драйвера
15) Загружается ОС.

## V. Расширенная настройка параметров работы

Конфигурирование в целом — это важнейший управляющий функционал среды ПНС. Благодаря ему становится возможной тонкая и точная настройка оборудования и функциональности ПНС. Конфигурирование обеспечивает параметрический доступ к отдельным составляющим системы, что облегчает разработку и упрощает наладку. Как было показано в статье, время перезапуска напрямую зависит от аппаратной конфигурации. Для ускорения работы системы и/или оптимизации её работы на помощь пользователям приходит конфигурирование. К примеру, в качестве возможностей по настройке машин конфигуратор ПНС Эльбрус версии 6 позволяет:

1) Общие настройки ПНС:
   a) запретить/разрешить многопоточность.
   b) перейти в режим минимальной конфигурации.
   c) ПНС предоставляет возможность запрета/разрешения поддержки работы аппаратных интерфейсов и драйверов.
   d) сбросить/установить все настройки по умолчанию.
   e) ПНС предоставляет возможность запрета/разрешения поддержки работы аппаратных интерфейсов и драйверов.
   f) перейти в вербозный режим.
   g) ПНС предоставляет возможность устанавливать допустимые технологические паузы.
   h) ПНС позволяет оператору выбирать жёсткий диск, раздел, имя файла для загрузки операционной системы.
   i) ПНС позволяет оператору формировать командную строку перед запуском Linux.
   j) ПНС позволяет оператору отключить режим мультилинка.
   k) ПНС позволяет настроить режимы работы контроллеров IPCC3.
   l) ПНС автоматически снимает питание со сбойного оборудования.
   m) ПНС автоматически переходит в режим мультилинка при детектировании его аппаратного наличия.
   n) ПНС позволяет управлять работой системы охлаждения и многое другое.
2) Настройки ОЗУ:

a) ПНС поддерживает и автоматически определяет три типа модулей DDR: RDIMM, LRDIMM, UDIMM.
b) установить тонкие настройки для физических уровней DDR.
c) задать допустимые частоты работы контроллерам памяти.
d) ПНС позволяет включить/выключить из конфигурации указанные модули памяти в заданных слотах.

3) Настройки памяти:
   a) ПНС предоставляет четыре штатных конфигурации PCIMEM и DMA и одну произвольную пользовательскую.
4) Настройки центрального процессора
   a) назначить предельную частоту работы группе процессоров или конкретному процессору.
   b) ввести или вывести из конфигурации любые процессоры и ядра процессоров.
   c) ПНС дает возможность ввести/вывести из конфигурации указанные южные мосты.
5) Настройки безопасности
   a) ПНС автоматически переходит в режим АПМДЗ [11] при детектировании его аппаратного наличия.
   b) ПНС позволяет отказаться от работы в режиме АПМДЗ.

## VI. Результаты замеров

Требуется сопоставить длительности загрузки ОС на весьма разных платформах. Исходя из этого, время перезапуска необходимо определять, фиксируя интервал от включения питания до передачи управления ОС. Для Windows — это конец фазы OS Loader [12]. Для Linux — старт самораспаковки. При всех прочих различиях в этой точке закончился POST, считан стартовый код ОС и подняты нужные для запуска системы компоненты, включая драйвер видеокарты. Результаты будем поверять средствами анализа производительности eventviewr и журнала событий в Windows, а в Linux утилита systemd-analyze.

В табл. 1 представлены результаты времени загрузки оборудования до старта ОС. В строке Эльбрус E2C3* представлен результат, полученный с использованием FRAM. Как видно из таблицы, даже при инициализации ОС с NVME, более высоких частотах процессора, меньшем числе южных мостов и равных объемах ОЗУ, скорость загрузки Intel не превосходит по времени ПНС Эльбрус. Конечно на другом оборудовании или при смене настроек результаты могут быть иные.

Таблица 1

*Время загрузки до старта ОС на различном оборудовании*

| Архитектура | Число ЦП | Число ядер | Макс. частота ЦП, ГГц | Число южных мостов | Память ОЗУ, ГБ | Тип диска | ОС | Время работы ПНС, сек |
|---|---|---|---|---|---|---|---|---|
| Эльбрус E2C3 | 1 | 2 | 2,0 | 1 | 8 | NVME | Linux 5.4.0-3.12-e2c3 | 7 |
| Эльбрус E2C3 | 1 | 2 | 2,0 | 1 | 16 | HDD | Linux 5.4.0-3.12-e2c3 | 8 |
| Эльбрус E2C3* | 1 | 2 | 2,0 | 1 | 16 | HDD | Linux 5.4.0-3.12-e2c3 | 7.6 |
| Intel Xeon E5607 | 1 | 4 | 2,27 | 1 | 2 | SAS | Windows 7 | 8 |
| Intel Core i7 4700HQ | 1 | 4 | 3,4 | 1 | 16 | SATA SSD | Windows 10 | 8 |
| Intel Core i5 11400F | 1 | 5 | 4,4 | 1 | 16 | NVME | Windows 10 | 10 |
| Intel Core i7 7500U | 1 | 2 | 3,5 | 1 | 16 | SATA SSD | Windows 10 | 11 |
| Эльбрус E16C | 1 | 16 | 2,0 | 1 | 64 | HDD | Linux 4.19.0-0-22520-e16c | 12 |
| Intel Core i5 11400 | 1 | 6 | 4,4 | 1 | 32 | NVME | Windows 11 | 12 |
| Intel Xeon E5 2683 v3 | 1 | 14 | 3,0 | 1 | 32 | HDD | Ubuntu 16.04 | 14 |
| Intel Core i7 720QM | 1 | 4 | 2,8 | 1 | 4 | HDD | Windows 7 | 15 |
| Эльбрус E16C | 2 | 32 | 2,0 | 2 | 128 | HDD | Linux 5.4.129-0-rt61-e16c | 32 |
| Intel Xeon X5570 | 2 | 8 | 3,3 | 1 | 128 | HDD | Linux 2.6.16.60-0.34-smp | 35 |

## VII. Заключение

Время загрузки зависит от аппаратной конфигурации. Объем оперативной памяти является значимым фактором. При прочих равных характеристиках оборудования разница становится заметна при объеме ОЗУ выше 16GB и существенно возрастает при 64GB и больше. Увеличение числа ядер процессора и южных мостов замедляет загрузку. Частота процессора и особенно скорость дисков существенно влияют на время. При использовании SSD перезапуск происходят в среднем ~1 секунду быстрее. Представленные данные времени загрузки могут быть сокращены при параллельной загрузке драйверов ПНС и сокращении аппаратной задержки. Ожидаемый выигрыш тут ещё 2-3 секунды.

## Литература


1. Chen Sha, Zhu-ying Lin, Design Optimization and Implementation of Bootloader in Embedded System Development, Software institute, Beijing University of Technology Beijing Engineering Research Center for IoT Software and SystemsBeijing, China, 2015
2. Угрюмов Е. П., Цифровая схемотехника, издательство «БХВ-Петербург», Санкт-Петербург, Россия, 2007
3. URL: https://www.denx.de/project/u-boot/ (26.04.2022)
4. S. Patton, Designing Embedded Quick Start Systems, Texas Instruments, URL: https://training.ti.com/sites/default/files/docs/Design-Quick-Start-Embedded-Systems-SLIDES_1.pdf (26.04.2022)
5. Y.A. Nedbailo, I. N. Bychkov, P. A. Chuchko, E. G. Panchenko, M. V. Slesarev, A. I. Troosh, V. M. Feldman, Elbrus-2C3: a Dual-Core VLIW Processor with Integrated Graphics, INEUM im.I.S.Bruka, Moscow, Russia, MCST JSC Moscow, Russia, 2021 год..
6. URL: https://safespi.org/ (25.04.2022)
7. Yinghai Lu, Li-Ta Lo, Gregory R. Watson, Ronald G. Minnich, CAR: Using Cache as RAM in LinuxBIOS, Advanced Computing Laboratory Los Alamos National Laboratory Los Alamos, New Mexico, USA, 2007
8. URL: https://www.i2c-bus.org/ (25.04.2022)
9. Скотт Мюллер, Модернизация и ремонт ПК, издательство «QUE», Идианаполис, Индиана, 2007 г.
10. URL: https://pcisig.com/ (25.04.2022)
11. Петров А.А., Компьютерная безопасность. Криптографические методы защиты. издательство «ДМК», Москва, Россия, 2008
12. URL: https://docs.microsoft.com/en-us/windows/client-management/advanced-troubleshooting-boot-problems (20.07.2022)


# Methods for fast implementation of loading in computing systems


Bychkof I. N., Lomako S.G., Lomako G.A.

INEUM, named I. S. Brook, Moscow, Russia, bignat@mail.ru



***Abstract* — this article discusses methods to reduce to a few seconds the boot time from power on to the start of the initialization of the operating system. The considered methods are universal, and some of them require hardware support. These methods have been tested in computing systems based on Elbrus processors and their efficiency has been analyzed.**



***Keywords* — fast boot, bootloader, quick start, boot-time optimization, multiple-resource management, optimization**


## References


1. Chen Sha, Zhu-ying Lin, Design Optimization and Implementation of Bootloader in Embedded System Development, Software institute, Beijing University of Technology Beijing Engineering Research Center for IoT Software and SystemsBeijing, China, 2015
2. Ugrumov E. P., Tsifrovaya skhemotekhnika (Digital circuitry), publishing house "BHV-Petersburg", St. Petersburg, Russia, 2007
3. URL: https://www.denx.de/project/u-boot/ (26.04.2022)
4. S. Patton, Designing Embedded Quick Start Systems, Texas Instruments,URL: https://training.ti.com/sites/default/files/docs/Design-Quick-Start-Embedded-Systems-SLIDES_1.pdf (26.04.2022).
5. Y.A. Nedbailo, I. N. Bychkov, P. A. Chuchko, E. G. Panchenko, M. V. Slesarev, A. I. Troosh, V. M. Feldman, Elbrus-2C3: a Dual-Core VLIW Processor with Integrated Graphics, INEUM im.I.S.Bruka, Moscow, Russia, MCST JSC Moscow, Russia, 2021 год..
6. URL: https://safespi.org/ (25.04.2022)
7. Yinghai Lu, Li-Ta Lo, Gregory R. Watson, Ronald G. Minnich, CAR: Using Cache as RAM in LinuxBIOS, Advanced Computing Laboratory Los Alamos National Laboratory Los Alamos, New Mexico, USA, 2007
8. URL: https://www.i2c-bus.org/ (25.04.2022)
9. Scot Mueller, Upgrading and Repairing PCs, Seventeenth Edition, publishing house «QUE», publishing house, Indianapolis, Indiana, USA 2007
10. URL: https://pcisig.com/ (25.04.2022)
11. Petrov A.A., Komp′yuternaya bezopasnost′ Kriptograficheskiye metody zashchity (Computer security Cryptographic methods of protection), publishing house «DMK», Moscow, Russia, 2008
12. URL: https://docs.microsoft.com/en-us/windows/client-management/advanced-troubleshooting-boot-problems (20.07.2022)