DOI: 10.58168/MoInSyTe2024_582-589
УДК 519.713.1

ПРОГРАММНЫЕ ИНСТРУМЕНТЫ С ОТКРЫТЫМ КОДОМ ДЛЯ ПРОЕКТИРОВАНИЯ ПРОЦЕССОРНЫХ ЯДЕР RISC-V

А.В. Строгонов[1], О.Л. Бордюжа[2], А.И. Строгонов[3]

[1]ФГБОУ ВО «Воронежский государственный технический университет
[2]ФГБОУ ВО «Воронежский государственный лесотехнический университет имени Г.Ф. Морозова»
[3]ФГБОУ ВО «Воронежский государственный университет»

Аннотация. В статье рассматривается международный опыт в разработке процессорных ядер на примере RISC-V с использованием программных инструментов с открытым кодом. Описано использование языка аппаратного конструирования Chisel в академических научных центрах за рубежом для создания библиотек сложных генераторов процессорных ядер, кэшей различных уровней, интерфейсов шин и других функциональных блоков в контексте систем на кристалле (СнК).

Ключевые слова: процессорные ядра, программные инструменты, функциональные блоки, эффективность, методология.

OPEN SOURCE SOFTWARE TOOLS FOR DESIGNING RISC-V PROCESSOR CORES

A.V. Strogonov[1], O.L. Bordyuzha[2], A.I. Strogonov[3]

[1]Voronezh State Technical University
[2]Voronezh State University of Forestry and Technologies named after G.F. Morozov
[3]Voronezh State University

Abstract. The article discusses international experience in the development of processor cores using the example of RISC-V using open source software tools. The use of the Chisel hardware design language in academic research centers abroad is described to create libraries of complex generators of processor cores, caches of various levels, bus interfaces and other functional blocks in the context of systems on a chip (SoC).

Keywords: processor cores, software tools, functional blocks, efficiency, methodology.

В настоящее время за рубежом в академических научных центрах широко используют язык аппаратного конструирования Chisel для создания библиотеки сложных генераторов процессорных ядер, кэшей различных уровней, интерфейсов шин (внутрикристальных межсоединений для соединений и управления функциональными блоками в СнК-разработках) и др.

Rocket Chip - это СнК-генератор разработанный в Калифорнийском университете в Беркли. Rocket Chip имеет открытый исходный код и доступно по лицензии BSD на Github. Rocket Chip поддерживает интеграцию пользовательских ускорителей в виде расширений набора команд, сопроцессоров или полностью независимых новых ядер. Процессорные ядра, сгенерированные Rocket были изготовлены в кремнии и показали способность загружать ОС Linux.

Rocket Chip генерирует процессорные ядра общего назначения, использующие открытый RISC-V ISA, и обеспечивает как генерацию стандартных конвейерных скалярных ядер (Rocket in-order) с очередным, так и ядер с внеочередным исполнением команд (Berkeley Out-of-Order Machine, BOOM). BOOM является суперскалярным ядром RV64G [1].

Chisel (**Constructing Hardware in a Scala Embedded Language**) - язык описания аппаратуры, надстроенный над Scala, который транслируется в Verilog код. Chisel можно рассматривать как библиотеку для Scala.

Chisel это язык описания аппаратного обеспечения с открытым исходным кодом, используемый для описания цифровых логических схем и схем на уровне регистровых передач (hardware construction language, HCL). Chisel добавляет примитивы аппаратного обеспечения к языку программирования Scala, предоставляя разработчикам возможности современного языка программирования для написания сложных параметризуемых генераторов схем, которые создают синтезируемый Verilog код. Эта методология позволяет создавать повторно используемые компоненты и библиотеки, такие как очередь FIFO и арбитры в стандартной библиотеке Chisel, повышая уровень абстракции в дизайне при сохранении детального контроля.

Scala является преемником Java. Scala более объектно-ориентирован, чем Java, а также обладает возможностями функционального языка. Сочетание двух подходов делает программирование на Scala гибким, повышает эффективность кода и позволяет реализовывать нестандартные решения [2].

Из RTL- описания на Chisel строится аппаратный граф, который, в свою очередь, превращается в промежуточное описание на языке firrtl, а уже после встроенный интерпретатор генерит из firrtl код Verilog или C++.

Verilator имеет аналогичную или лучшую производительность по сравнению с Verilog симуляторами с закрытым исходным кодом (например, Carbon Design Systems Carbonator, Modelsim/Questa, Cadence Incisive/NC-Verilog, Synopsys VCS, VTOC и Pragmatic CVer/CVC).

Симулятор Verilator позволяет преобразовать Verilog модули в C++ классы, которые потом компилируются в обычную исполняемую программу. Что позволяет достичь очень высокой производительности. Verilator, может обрабатывать только синтезируемый Verilog код с учетом архитектурного базиса ПЛИС [3].

Verilator не переводит Verilog HDL напрямую на C ++ или SystemC. Verilator компилирует код пользователя в более быструю оптимизированную модель с разделением на потоки, которая, в свою очередь, помещается в модуль C ++/SystemC. Результатом является скомпилированная модель Verilog, которая выполняется даже в одиночном потоке более чем в 10 раз быстрее, чем standalone SystemC, а в одиночном потоке примерно в 100 раз быстрее, чем интерпретируемые симуляторы Verilog, такие как Icarus - Verilog. Дополнительное 2-10-кратное ускорение может быть получено за счет многопоточности (что дает в общей сложности 200-1000 раз больше по сравнению с интерпретируемыми симуляторами).

Verilator — это инструмент, который компилирует исходные коды Verilog и SystemVerilog в высокооптимизированный (и, возможно, многопоточный) циклически точный код C++ или SystemC. Преобразованные модули можно создавать и использовать в тестовом стенде C++ или SystemC для проверки и/или моделирования [4].

На рис. 1 показан пример экземпляра Rocket Chip. На нем изображены две ячейки (плитки) с процессорными ядрами Rocket и BOOM, которые подключены к 4 рядам кэша (специальный буфер быстрой памяти) L2$, который через интерфейс L2toIO подключен к внешним системам ввода-вывода и памяти с помощью моста TileLink/AXI4. Генерируемые процессорные ядра содержат L1-кэши инструкций и данных (L1I$ и L1D$), TLB (буферы быстрого перевода) и FPU (модуль операций с плавающей запятой) с интерфейсами MemIO и HostIO для связи с внешним миром, интерфейс "RoCC", который связывается с подключенными ядрами ускорителей/сопроцессоров.

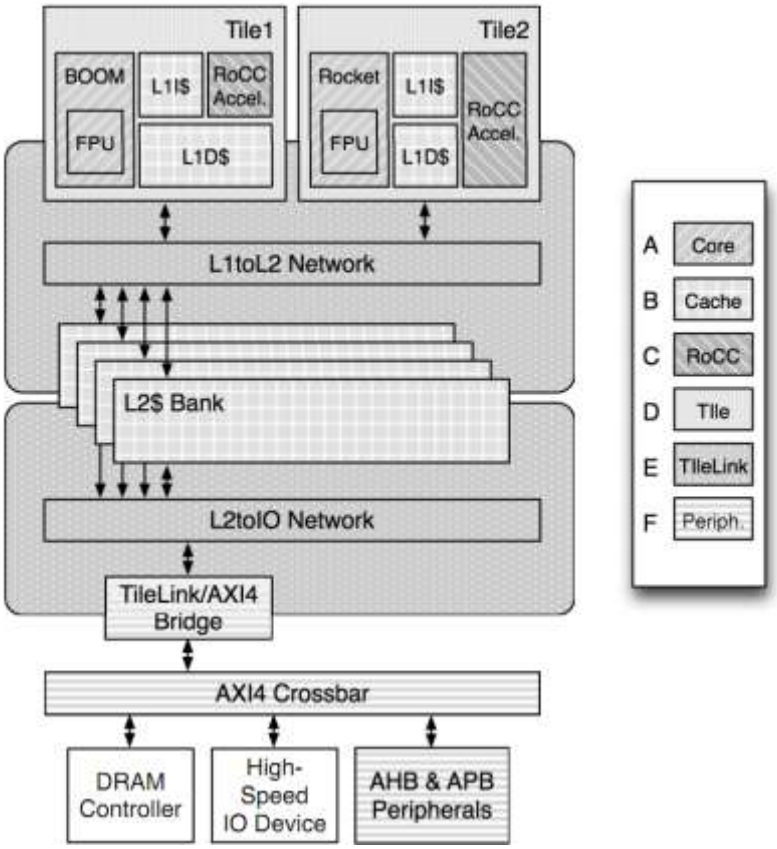

Рис. 1. Rocket Chip генератор микропроцессорных ядер состоит из следующих подкомпонентов: основной генератор ядер (A); генератор кэша (B); генератор RoCC-совместимого сопроцессора (C); генератор ячеек (D); TileLink – генератор связей между ячейками (E); периферийные устройства (F)

Генераторы ядер Rocket и BOOM могут включать в себя: дополнительные модули FPU, настраиваемые конвейеры функциональных блоков и настраиваемые предсказатели ветвлений программы; семейство генераторов кэша и TLB с настраиваемыми размерами, ассоциативностью и политики замены; пользовательский интерфейс сопроцессора RoCC; шаблон генератора ячеек, с учетом когерентности кэшей; генератор связей между плитками (генератор сетей агентов когерентности кэша и связанных с ним контроллеров кэша). Опции конфигурации включают: количество ячеек, политику согласованности, наличие общего резервного хранилища и реализация базовых физических сетей); периферийные устройства: генераторы шин, совместимых с AMBA (AXI, AHB-Lite и APB) и множество конверторов и контроллеров, включая процессор Z-scale.

Rocket Chip представляет собой генератор скалярных ядер RV64G с 5-ступенчатым конвейером (выборка, декодирование, выполнение, память, обратная

запись), с модулем управления памятью MMU, которое поддерживает виртуализацию памяти на основе страниц, неблокирующий кэш данных и интерфейс с предсказанием переходов [5].

Предсказание ветвлений программы настраивается и обеспечивается целевым буфером переходов (BTB), таблицей истории переходов (BHT) и стеком адресов возврата (RAS). Доступна поддержка некоторых расширений ISA (M, A, F, D). BOOM поддерживает спекулятивное выполнение ветвлений программы и предсказание переходов, используя BTB, RAS и параметризированный резервный предиктор (BPD).

Некоторые из доступных поддерживаемых предикторов, которые могут быть созданы, включают gshare предиктор (двухуровневая адаптивная схема предсказания ветвлений, с использованием xor, называется gshare) и TAGE предиктор (тегированный геометрический, TAgged Geometric предиктор с размером истории, разновидность которого используется в процессоре с архитектурой Intel Haswell). BOOM использует модуль загрузки/накопления операций с агрессивным планированием. На рис. 2 показан сгенерированный суперскалярный RV64G процессор BROOM.

Rocket Chip реализуется на Xilinx Ultrascale+ FPGA серия Zynq-7000 с использованием инструментов разработки и отладки встраиваемых микропроцессорных систем таких как Parallella и ZedBoard. Реализация ядра RV32I возможна на ПЛИС серии Lattice ICE40. Код написан на SystemVerilog и компилируется при помощи IceStorm.

Китайская академия наук придаёт большое значение созданию экосистем технологий с открытым исходным кодом. В 2018 году был сформирован Китайский альянс RISC-V для развития проектов. Процессор Xiangshan и разработанная к нему ОС Aolai демонстрируют стремление Китая укрепить экосистему RISC-V.

Институт компьютерных технологий китайской академии наук в 2020 г представил проект XiangShan, развивающий высокопроизводительный открытый процессор на базе архитектуры набора команд RISC-V (RV64GC) с использованием программных инструментов с открытым исходным кодом для их проектирования. Наработки проекта открыты под пермиссивной лицензией (лицензии на программное обеспечение, которые практически не ограничивают свободу действий пользователей ПО и разработчиков, работающих с исходным кодом) MulanPSL 2.0.

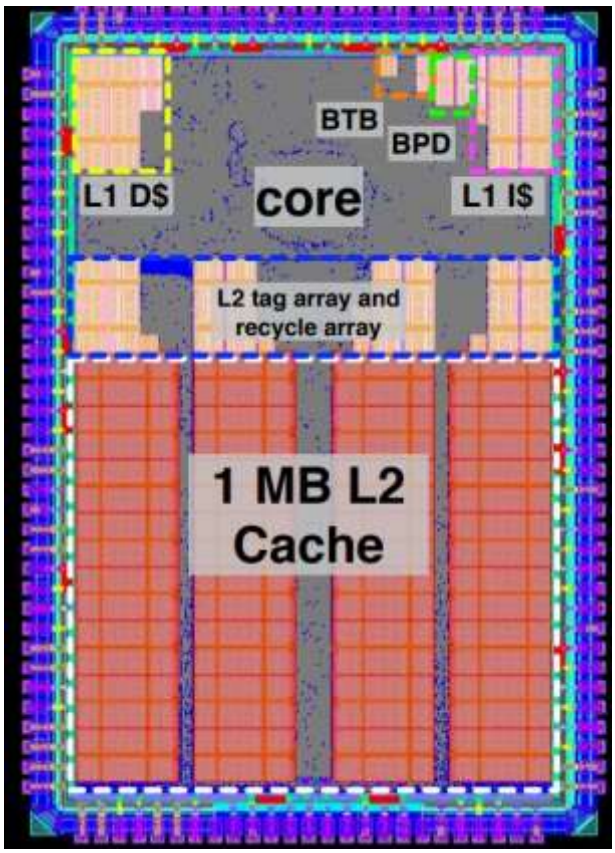

Рис. 2. Топологическое планирование кристалла суперскалярного RV64G процессора BROOM, изготовленного на кремниевой фабрике TSMC по технологии 28 нм

В настоящее время разработаны 3 поколения процессоров (описание аппаратных блоков процессоров RISC-V представлено на языке Chisel). Микропроцессорные ядра проекта XiangShan были реализованы в базисе ПЛИС и проведено их моделирование с использованием Verilog-симулятора Verilator с открытым исходным кодом [6]. В качестве эталонной операционной системы, используемой при тестировании реализации на базе ПЛИС, применяется Debian GNU/Linux.

Компания Alibaba Cloud Intelligence (китайский облачный провайдер) открыла исходные коды, а также соответствующее программное обеспечение и инструменты четырех процессорных ядер RISC-V: XuanTie E902, E906, C906 и C910. Хотя RISC-V является открытым стандартом и существует значительная доля ядер RISC-V с открытым исходным кодом, многие коммерческие ядра RISC-V имеют закрытый исходный код.

На базе этих ядер можно самостоятельно разработать свой процессор (СнК) и начать массовое производство. Открытый доступ к документации ядра

процессора облегчает разработку программного обеспечения и портирование необходимых драйверов.

Схемы, описания аппаратных блоков на языке Verilog, симулятор и сопутствующая проектная документация опубликованы на GitHub под лицензией Apache 2.0. Отдельно опубликованы адаптированные для работы с процессорами XuanTie версии компиляторов GCC и LLVM, библиотека Glibc, инструменты Binutils, загрузчик U-Boot, ядро Linux, middleware с интерфейсом OpenSBI (Supervisor Binary Interface), платформа для создания встраиваемых систем на базе Linux Yocto Project. Ядра OpenE902, OpenE906, OpenC906 и OpenC910, доступны на GitHub под лицензией Apache 2.0.

Как видим, многие китайские проекты ставят перед собой задачу разработать процессорные ядра RISC-V ISA с использованием программных инструментов САПР с открытым исходным кодом.

Список литературы

1. Фролов В.А. Исследование технологии RISC-V / В.А. Фролов, В.А. Галактионов, В.В. Санжаров // Труды ИСП РАН, том 32, вып. 2, 2020 г., стр. 81-98. DOI: 10.15514/ISPRAS–2020–32(2) –7.
2. Харрис С.Л. Цифровая схемотехника и архитектура компьютера: RISC-V / Сара Л. Харрис, Дэвид Харрис. // пер. с англ. В.С. Яценкова, А.Ю. Романова; под ред. А. Ю. Романова. – М.: ДМК Пресс, 2021. – 810 с.
3. Строгонов А.В. Эффективный подход в разработке управляющих автоматов микропроцессорных ядер ПЛИС /А.В. Строгонов, О.Л. Бордюжа, А.И. Строгонов // Электроника: Наука, Технология, Бизнес, 2024. № 1. С. 1-8.
4. Ачкасов А.В. Особенности проектирования микросхем, выполненных по глубоко-субмикронным технологиям / А.В. Ачкасов, М.В. Солодилов, Н.Н. Литвинов, П.А. Чубунов, В.К. Зольников, Д.В. Шеховцов, О.Л. Бордюжа // Моделирование систем и процессов. – 2022. – Т. 15, № 4. – С. 7-17.
5. Строгонов А.В. Схемы очистки конфигурационной памяти ПЛИС / А.В. Строгонов, О.И. Орехов, А.С. Плохих, О.Л. Бордюжа // Твердотельная электроника, микроэлектроника и наноэлектроника. Межвузовский сборник научных трудов. – Воронеж, 2023. - С. 279-281.
6. Зольников В.К. Обзор программ для САПР субмикронных СБИС и учет электрофизических эффектов глубоко субмикронного уровня / В.К. Зольников, А.Л. Савченко, А.Ю. Кулай // Моделирование систем и процессов. – 2019. – Т. 12, № 1. – С. 40-47.

References

1. Frolov V.A. Research of RISC-V technology / V.A. Frolov, V.A. Galaktionov, V.V. Sanzharov // Proceedings of ISP RAS, volume 32, issue. 2, 2020, pp. 81-98. DOI: 10.15514/ISPRAS–2020–32(2)–7.

2. Harris S.L. Digital Circuit Design and Computer Architecture: RISC-V / Sarah L. Harris, David Harris. // lane from English V.S. Yatsenkova, A.Yu. Romanova; edited by A. Yu. Romanova. – M.: DMK Press, 2021. – 810 p.

3. Strogonov A.V. An effective approach to the development of control machines for FPGA microprocessor cores / A.V. Strogonov, O.L. Bordyuzha, A.I. Strogonov // Electronics: Science, Technology, Business, 2024. No. 1. P. 1-8.

4. Achkasov A.V. Features of the design of microcircuits made using deep-submicron technologies / A.V. Achkasov, M.V. Solodilov, N.N. Litvinov, P.A. Chubunov, V.K. Zolnikov, D.V. Shekhovtsov, O.L. Bordyuzha // Modeling of systems and processes. – 2022. – T. 15, No. 4. – P. 7-17.

5. Strogonov A.V. Schemes for clearing FPGA configuration memory / A.V. Strogonov, O.I. Orekhov, A.S. Plokhikh, O.L. Bordyuzha // Solid-state electronics, microelectronics and nanoelectronics. Interuniversity collection of scientific papers. – Voronezh, 2023. - pp. 279-281.

6. Zolnikov V.K. Review of programs for CAD submicron VLSI and taking into account electrophysical effects at the deep submicron level / V.K. Zolnikov, A.L. Savchenko, A.Yu. Kulai // Modeling of systems and processes. – 2019. – T. 12, No. 1. – P. 40-47.