



Санкт-Петербургский государственный университет
Кафедра системного программирования

Интеграция SCR1 в платформу LiteX для запуска на Colorlight i9

МУКОВЕНКОВ Роман Сергеевич, группа 23.Б11-мм

Научный руководитель: ст. преподаватель кафедры ИАС, Смирнов К. К.

Санкт-Петербург
2025

- Применение FPGA для верификации ASIC
- Использование HDL для описания аппаратных систем с возможностью перепрограммирования
- Преимущества: высокая производительность благодаря параллелизму, конвейеризации и минимальным накладным расходам
- Популярность RISC-V архитектур на HDL и возможность коммерческого использования SCR1 благодаря открытой лицензии Solderpad Hardware

- Существует четыре отлаженных СнК от компании Syntacore и один от Луконенко Никиты Игоревича, студента третьего курса кафедры системного программирования
- Были созданы с помощью ручного или полуручного подбора компонентов: контроллера памяти, мостов между шинами, UART модуль
- Используют написанный Syntacore bootloader: sc-bl
- А что если нам хочется полностью автоматизировать для всех FPGA «в один клик»? Для этого нам нужен LiteX

Преимущества LiteX перед ручным методом

- Написан на Python, имеет библиотеку компонентов, как у проприетарных инструментов вендоров
- Можно описывать свой SoC на Python и добавлять компоненты из этой библиотеки
- Далее, процессор и FPGA абстрагируются
- Поддержка порядка 198 плат, абстрагированных в LiteX, при успешной абстракции процессора
- Будем писать часть, отвечающую за абстракцию ресурсов процессора SCR1, проверим запуском на Colorlight i9

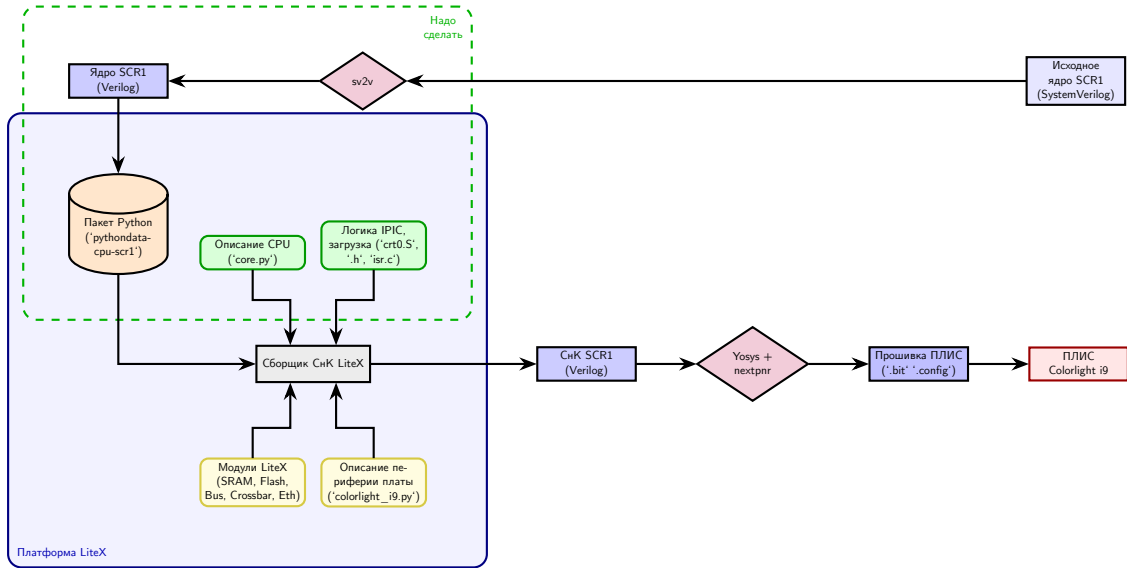
Целью работы является запуск ядра SCR1 на плате Colorlight i9

Задачи:

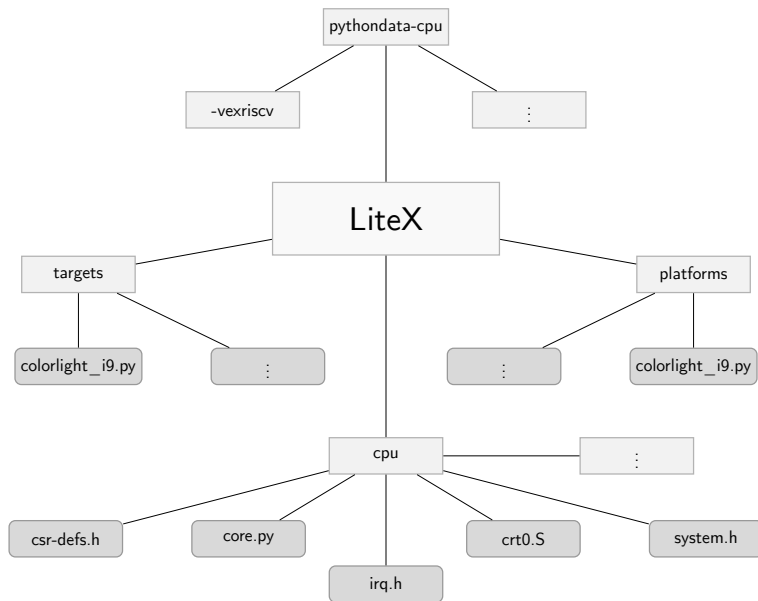
- Изучить работу платформы LiteX в объёме, необходимом для успешной интеграции SCR1
- Рассмотреть работу уже интегрированного в платформу LiteX ядра VexRiscv для проверки взаимодействия Colorlight i9 и платформы LiteX
- Реализовать описание SCR1 в LiteX в соответствующем Python файле, подготовить исходные файлы для синтеза
- Интегрировать базовую версию обработчика прерываний, называемого IPIC, в LiteX

В конце мы запустим SCR1 с LiteX firmware!

Что требуется сделать



LiteX для абстрагирования процессоров



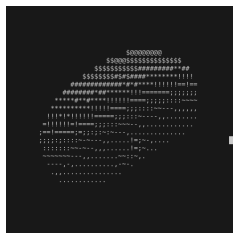
Какие инструменты доступны?

- Плата Colorlight i9 содержит ECP5 LFE5U-45F от Lattice.
- Выбран open-source toolchain (Yosys + nextpnr) ввиду его доступности.



Демо на VexRiscv

- Для проверки работы платформы для Colorlight i9 синтезировано и отправлено ядро VexRiscv
- Команды:
 - ▶ `litex colorlight_i9.py --build` – сборка битстрима
 - ▶ `litex colorlight_i9.py --load` – загрузка через JTAG
 - ▶ `litex_term /dev/ttyUSBX` – терминал взаимодействия через UART
 - ▶ `litex_term /dev/ttyUSBX --kernel=<file>.bin` – послать firmware через UART
- Результат: светодиод мигает, через UART появляется меню LiteX BIOS, доступны команды (Hello World на языке C, анимация ASCII-«пончика», управление LED)



Подготовка исходных файлов

- Yosys не полностью поддерживает SystemVerilog, в отличие от Verilog!
- Попытка синтезировать исходные файлы с ограниченной поддержкой вне LiteX
 - ▶ Заменить синтаксический сахар (return заменяет присвоение значения имени функции и завершает её)
 - ▶ Убрать косметику (endfunction : <название_функции>)
 - ▶ Вставить «неконстантные» параметры (использование в range)
- struct packed вне собственного модуля выдавало непонятную ошибку, связанную с range.
 - ▶ Были собраны исходные файлы Yosys с отладочными сообщениями, чтобы понять, почему какой-то assert падал
 - ▶ К сожалению, typedef struct packed не поддерживались вне своего модуля. Менять логику было крайне проблематично. О проблеме известно, но решение ломает другое использование <https://github.com/YosysHQ/yosys/issues/4653>
- Был опробован synlig - другой парсер для Yosys, но ничего не определялось из include файлов
- Был выбран инструмент sv2v, который транслировал SystemVerilog в Verilog файлы.
- Транслированные файлы переносились в папку pythondata-cpu-scr1, где устанавливались с помощью pip install .

- Структура класса процессора:
 - ▶ Наследование от класса LiteX CPU
 - ▶ Архитектура rv32i, m и с extensions
- Подключение шин AXI:
 - ▶ Две независимые шины для инструкций и данных
 - ▶ Сопряжение сигналов AXI:
 - ★ AW/AR/W/B/R каналы
 - ★ Поддержка burst-транзакций
 - ▶ К сожалению, синтезировать прямую поддержку AXI не получилось
 - ▶ Кроме того, она всё равно использует более простой AXI-Lite
 - ▶ Поэтому в SCR1 LiteX используется Wishbone bridge
 - ▶ Параметр `–bus-bursting` ни на что не влиял, а `–bus-interconnect crossbar` помог и увеличить скорость, и уменьшить площадь
- Память:
 - ▶ Адреса основных компонентов:
 - ★ `main_ram`: 0x10000000
 - ★ CSR и I/O: 0xff000000
 - ★ ROM: 0xffffc0000 (адрес сброса)
 - ★ SRAM: 0xffff00000
 - ★ TCM: 0xf0000000

Сравнение использования ресурсов

Ресурс	VexRiscv	SCR1	FazyRV	picorv32	firev	Colorlight i9
DP16KD	49 (45%)	100 (92%)	39 (36%)	41 (37%)	39 (36%)	108 (100%)
TRELLIS_FF	2884 (6%)	4138 (9%)	1771 (4%)	2621 (5%)	2067 (4%)	43848 (100%)
TRELLIS_COMB	6465 (14%)	17407 (39%)	4167 (9%)	6442 (5%)	5351 (12%)	43848 (100%)

Таблица: Сравнение использования ресурсов для SCR1 и др. процессоров

Интеграция обработчика прерываний IPIC

- SCR1 использует IPIC (Integrated Programmable Interrupt Controller) для обработки прерываний
- LiteX не имеет реализации IPIC, её надо писать самому. Но интерфейс простой!
- Что было сделано:
 - ▶ fork Zephyr RTOS от Syntacore имеет реализацию интерфейса IPIC, также пришлось смотреть в спецификацию
 - ▶ Инициализация связывает провод-линии прерываний с логическим представлением в таблице — вектором
 - ▶ LiteX автоматически заполняет таблицу прерываний, таймер пока не поддерживаем: в mie 0x800
 - ▶ mstatus.MIE включает глобальные прерывания, при нажатии выполнение переходит в адрес в mtvec
 - ▶ Сохранение регистров, вызов isr, который берёт вектор из soi и выполняет по этому индексу необходимую функцию
- Результат: UART на FPGA работает, запуск программы с пончиком после некоторых изменений тоже

Поверхностное сравнение скоростей процессоров

Параметр	VexRiscv	SCR1	FazyRV	picorv32	firev
Желаемая частота, МГц	60	60	60	60	60
Частота критического пути, МГц	45.32	23.13	47.93	51.27	54.7
Послед. чт. 2 МиБ SDRAM, МиБ/с	30.2	17.2	10	17.5	24.5
Послед. зап. 2 МиБ SDRAM, МиБ/с	22.1	13.1	8.3	12.5	16.2
Послед. чт. 4 КиБ Flash, МиБ/с	2.8	2.6	2.6	1.7	3.1
Случ. чт. flash 4 КиБ, МиБ/с	1.3	0.768	0.046	0.769	0.17
Скорость пончика	Быстрая	Средняя	Оч. медл.	Медл.	Оч. медл

Таблица: Сравнение скорости SCR1 (interconnect) с др. процессорами

- По адресу 0xf0000000 лежит TCM, 64 КиБ, последовательная скорость записи и чтения 4 КиБ 33.6 МиБ/сек и 33.6 МиБ/сек соответственно
- В неё можно поместить SRAM для повышенной скорости работы

В результате работы было сделано:

- Изучена работа платформы LiteX в объёме, необходимом для успешной интеграции SCR1
- Рассмотрена работа уже интегрированного в платформу LiteX ядра VexRiscv для проверки взаимодействия Colorlight i9 и платформы LiteX
- Реализовано описание SCR1 в LiteX в соответствующем Python файле, подготовлены исходные файлы для синтеза
- Интегрирована базовая версия обработчика прерываний, называемого IPIC, в LiteX

Цель выполнена, SCR1 запущено на Colorlight i9

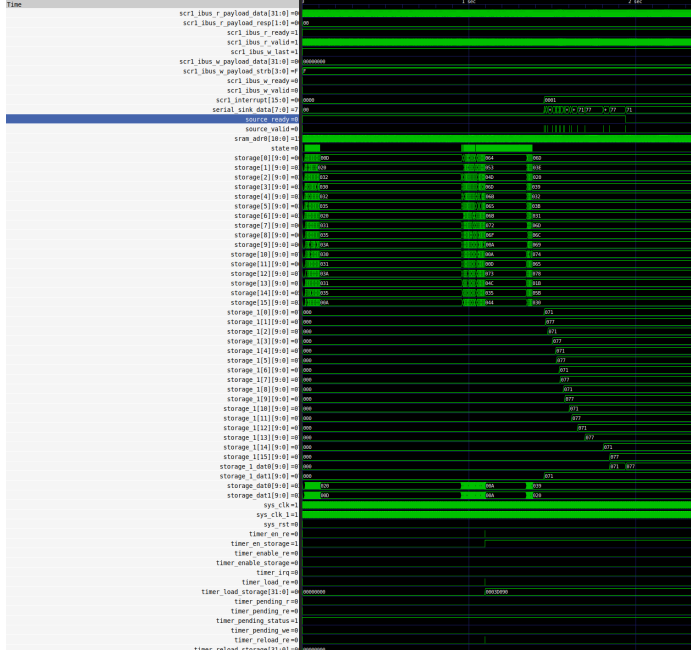


Рис.: Waveform, сгенерированная LiteX


```

INFO:SoC:
INFO:SoC:
INFO:SoC:
INFO:SoC:
INFO:SoC: Build your hardware, easily!
INFO:SoC:-----
INFO:SoC:Creating SoC... (2025-03-09 17:21:49)
INFO:SoC:-----
INFO:SoC:FPGA device : LFESU-45F-6BG3B1C.
INFO:SoC:System clock: 60.00MHz.
INFO:SoCBushHandler:Creating Bus Handler...
INFO:SoCBushHandler:32-bit wishbone Bus, 4.0GiB Address Space.
INFO:SoCBushHandler:Adding reserved Bus Regions...
INFO:SoCBushHandler:Bus Handler created.
INFO:SoCCSRHandler:Creating CSR Handler...
INFO:SoCCSRHandler:32-bit CSR Bus, 32-bit Aligned, 16.0KiB Address Space, 2048B Paging, big Ordering (Up to 32 Locations).
INFO:SoCCSRHandler:Adding reserved CSRs...
INFO:SoCCSRHandler:CSR Handler created.
INFO:SoCIRQHandler:Creating IRQ Handler...
INFO:SoCIRQHandler:IRQ Handler (up to 32 Locations).
INFO:SoCIRQHandler:Adding reserved IRQs...
INFO:SoCIRQHandler:IRQ Handler created.
INFO:SoC:-----
INFO:SoC:Initial SoC:
INFO:SoC:-----
INFO:SoC:32-bit wishbone Bus, 4.0GiB Address Space.
INFO:SoC:32-bit CSR Bus, 32-bit Aligned, 16.0KiB Address Space, 2048B Paging, big Ordering (Up to 32 Locations).
INFO:SoC:IRQ Handler (up to 32 Locations).
INFO:SoC:-----
INFO:SoC:Controller ctrl added.
INFO:SoC:CPU scr1 added.
INFO:SoC:CPU scr1 adding IO Region 0 at 0xffff00000 (Size: 0x0f00000).
INFO:SoCRegion:Region size rounded internally from 0x0f000000 to 0x01000000.
INFO:SoCBushHandler:io0 Region added at Origin: 0xffff00000, Size: 0x0f000000, Mode: RW, Cached: False, Linker: False.
INFO:SoC:CPU scr1 overriding main_ram mapping from 0x40000000 to 0x10000000.
INFO:SoC:CPU scr1 overriding rom mapping from 0x00000000 to 0xffff0000.
INFO:SoC:CPU scr1 overriding sram mapping from 0x01000000 to 0xffffe000.
INFO:SoC:CPU scr1 setting reset address to 0xffffc000.
INFO:SoC:CPU scr1 adding Bus Master(s).
INFO:SoCBushHandler:cpu_bus0 Bus adapted from AXI 32-bit to Wishbone 32-bit.
INFO:SoCBushHandler:cpu_bus0 added as Bus Master.
INFO:SoCBushHandler:cpu_bus1 Bus adapted from AXI 32-bit to Wishbone 32-bit.
INFO:SoCBushHandler:cpu_bus1 added as Bus Master.
INFO:SoC:CPU scr1 adding Interrupt(s).
INFO:SoCBushHandler:rom Region added at Origin: 0xffffc000, Size: 0x00020000, Mode: RX, Cached: True, Linker: False.
INFO:SoCBushHandler:rom added as Bus Slave.
INFO:SoC:RAM rom added Origin: 0xffffc000, Size: 0x00020000, Mode: RX, Cached: True, Linker: False.
INFO:SoCBushHandler:sram Region added at Origin: 0xffffe000, Size: 0x00002000, Mode: RWX, Cached: True, Linker: False.
INFO:SoCBushHandler:sram added as Bus Slave.
INFO:SoC:RAM sram added Origin: 0xffffe000, Size: 0x00002000, Mode: RWX, Cached: True, Linker: False.
INFO:SoCIRQHandler:uart IRQ allocated at Location 0.
INFO:SoCIRQHandler:timer0 IRQ allocated at Location 1.
INFO:SoCBushHandler:Allocating Cached Region of size 0x00800000...
INFO:SoCRegion:Region size rounded internally from 0xffffffff to 0x10000000.
INFO:SoCBushHandler:spiflash Region allocated at Origin: 0x00000000, Size: 0x00800000, Mode: RW, Cached: True, Linker: False.
INFO:SoCBushHandler:spiflash added as Bus Slave.
INFO:SoCBushHandler:main_ram Region added at Origin: 0x10000000, Size: 0x00800000, Mode: RWX, Cached: True, Linker: False.
INFO:SoCBushHandler:main_ram added as Bus Slave.

```

Рис.: LiteX подобрал нужные компоненты: IRQ, CSR, Bus, ROM, SRAM, Interconnect

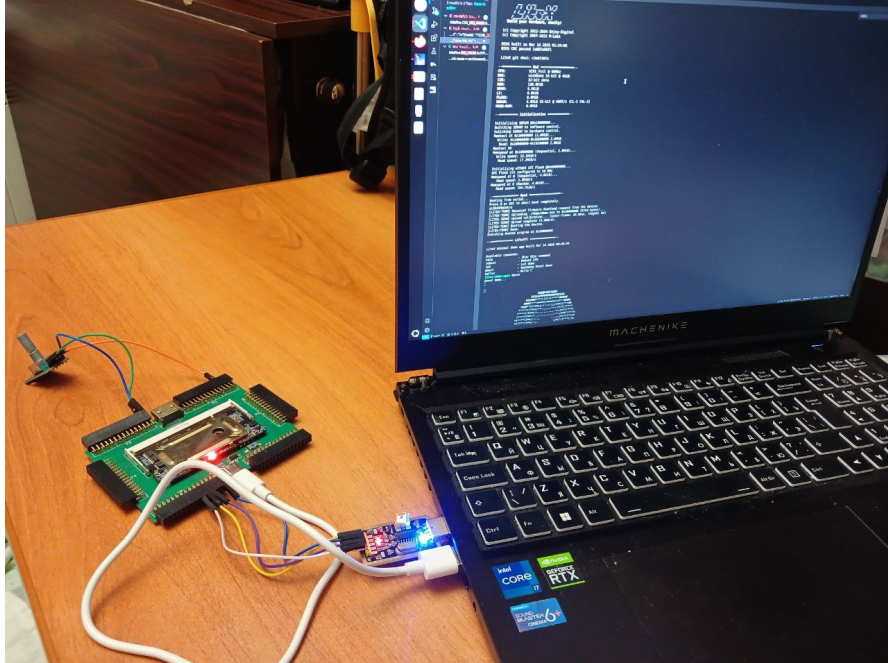


Рис.: Как настроена связь между компьютером и FPGA

