НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Московский институт электроники и математики им. А.Н. Тихонова
НИУ ВШЭ

Низкоуровневое моделирование сетей на кристалле

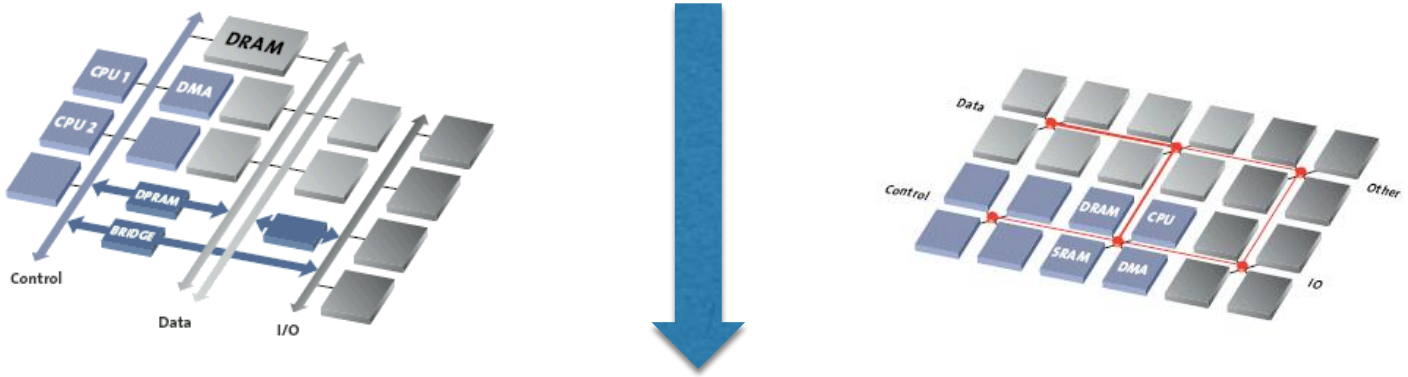
Студент ОП «ИВТ», 4 курс, группа БИВ 245 Грушевский Н.И.
Студент ОП «ИВТ», 4 курс, группа БИВ 245 Талибов С.М.

Научный руководитель, к.т.н., доц. ДКИ МИЭМ НИУ ВШЭ
Лежнев Е.В.

Москва, 2025

Сеть на кристалле

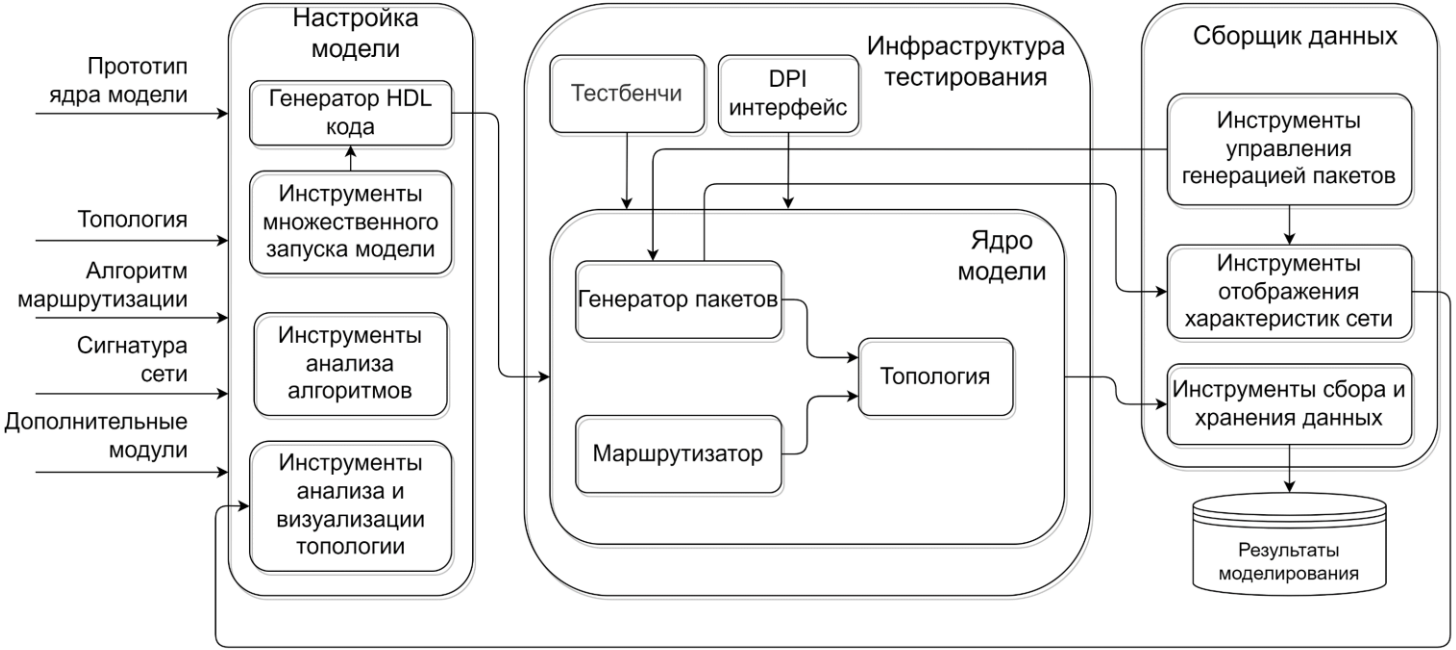
Система на кристалле (СнК) – электронная схема, выполняющая роль целого устройства, но размещенного на одной интегральной схеме. Элементы в схеме соединены с помощью высокопроизводительных шин (AMBA, Wishbone и др.).



Сеть на кристалле (СтнК) – многоядерная СнК, где роль вычислительных узлов выполняют вычислительные IP-блоки, которые связаны короткими соединениями для обмена информацией.

САПР низкоуровневого моделирования СтнК

HDLNocGen – САПР для низкоуровневого моделирования СтнК. Позволяет автоматизировать процесс разработки сети, подготовки к прототированию на ПЛИС и сбору характеристик работы.

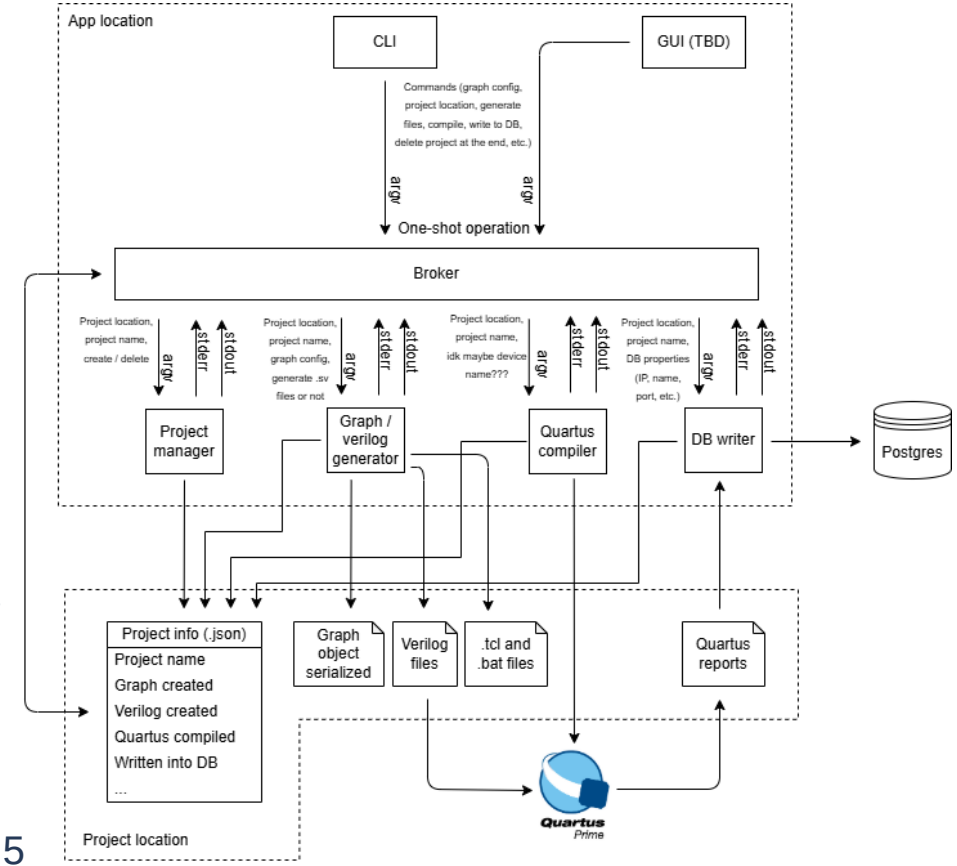


САПР низкоуровневого моделирования СтнК

Развитие HDLNoCGen:
2022 – диссетрация на соискание ученой степени кандидата технических наук Лежнева Е.В. «Автоматизация низкоуровневого моделирования сетей на кристалле»;
2023-2026 – проект №1718 «HDLNoCGen: САПР низкоуровневого моделирования сетей на кристалле»;
2025 – ВКР Жибуртович Е.М. «Графический конфигуратор сетей на кристалле»;
2025-2026 – проект №2078 «HDLNoCGen: Разработка графического конфигуратора сети на кристалле»;
2025-2026 – проект №2322 «HDLNoCGen: моделирование компонентов сетей на кристалле»;

HDLNoCGen – структура приложения

- СОА (Сервис-ориентированная архитектура) реализована на основе вызывающих друг друга исполняемых файлов;
- 4 сервиса, которые либо координируются брокером, либо ведут себя как независимые утилиты при их непосредственном вызове:
    - Project_manager – формирование проекта СтнК;
    - Graph_verilog_generator - создание объекта графа и HDL файлов;
    - Quartus_compiler – связь с САПР Quartus;
    - Database_writer – запись результатов компиляции в СУБД.



HDLNoCGen – возможности

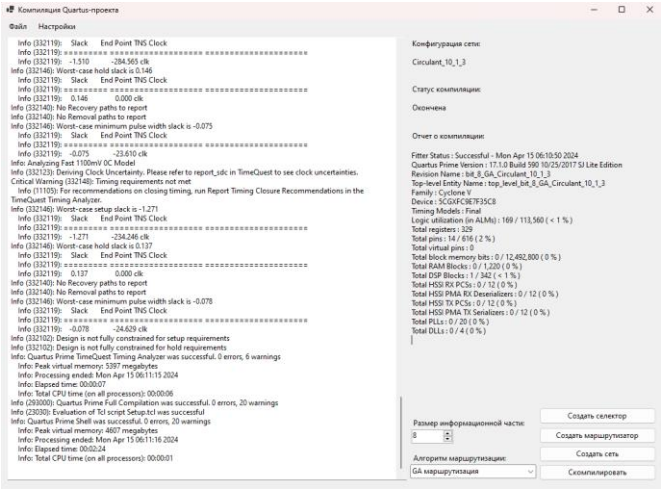
Автоматизация работы с САПР Quartus

Этапы взаимодействия с Quartus:

- Создание Verilog-файлов;
- Создание Tcl-скрипта;
- Запуск Tcl-скрипта, компиляция проекта;
- Считывание отчета. с

Возможности UI:

- Задание параметров маршрутизации и пакетов данных;
- Создание Verilog-файлов по-отдельности;
- Запуск компиляции;
- Сохранение вывода командной строки и отчета о компиляции в .txt файлы



HDLNoCGen – возможности

Работа с базой данных

Этапы выгрузки в базу данных:

- Считывание данных, полученных при высокоуровневом моделировании сети на кристалле, в специальную структуру;
- Парсинг отчета с последующей записью релевантных показателей в ту же структуру;
- Передача данных из структуры в базу данных;
- Структура собранных данных: тип топологии, сигнатура сети, параметры буфера пакетов, арбитр данных, размер пакета данных, занимаемые ресурсы чипа, характеристики работы сети, и пр.

topologies.csv 290.37 KiB

Open in Web IDE | Replace | Delete

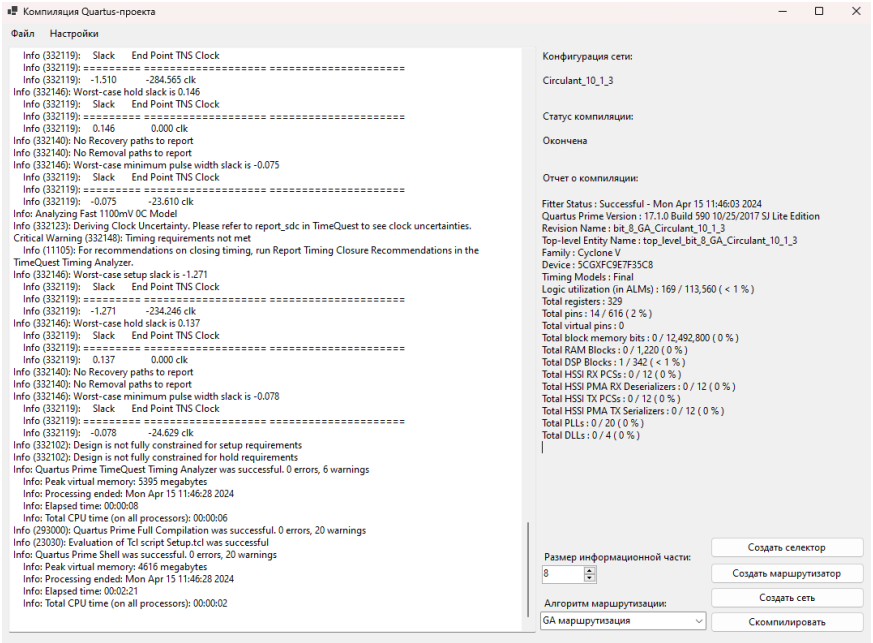
topology	graph_type	buffers	buffers_position	arbiter	algorithm	generators	info_length	alms	registers	frequency
Mesh	index	front	4	round_robin	xy	3, 3	8	1427	1940	143.7
Mesh	index	front	8	round_robin	xy	3, 3	8	2713	3615	118.34
Torus	index	front	8	round_robin	xy	3, 3	8	3804	4932	108.13
Mesh	index	front	16	round_robin	xy	3, 3	8	4784	6937	96.16
Mesh	index	front	24	round_robin	xy	3, 3	8	7423	10257	85.8
Mesh	index	front	32	round_robin	xy	3, 3	8	9611	13545	88.85
Mesh	index	front	64	round_robin	xy	5, 8	8	109569	163378	75.28
Torus	line	front	4	round_robin	xy	5, 8	8	10173	12811	109.27
Mesh	pointer	front	52	round_robin	xy	4, 4	8	25300	43444	78.59

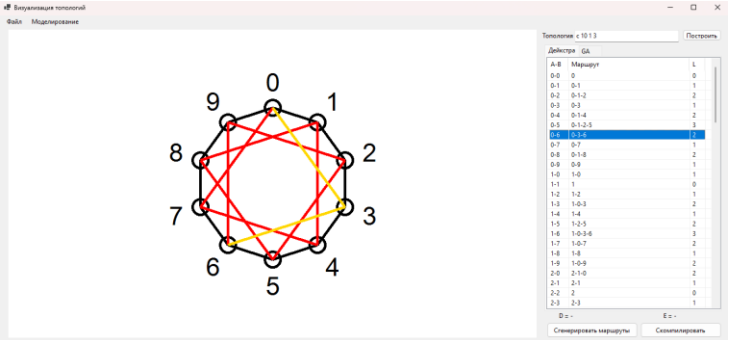
База данных

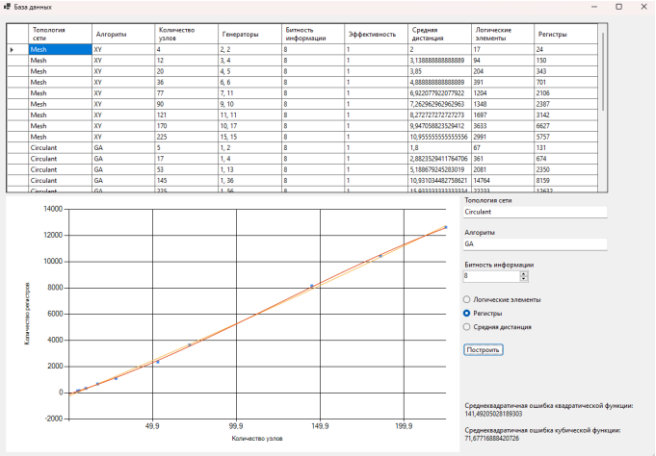
Топология сети	Алгоритм	Количество узлов	Генераторы	Битность информации	Эффективность	Средняя дистанция	Логические элементы	Регистры
Mesh	XY	4	2, 2	8	1	2	17	24
Mesh	XY	12	3, 4	8	1	3,138888888888889	94	150
Mesh	XY	20	4, 5	8	1	3,85	204	343
Mesh	XY	36	6, 6	8	1	4,888888888888889	391	701
Mesh	XY	77	7, 11	8	1	6,922077922077922	1204	2106
Mesh	XY	90	9, 10	8	1	7,262962962962963	1348	2387
Mesh	XY	121	11, 11	8	1	8,272727272727273	1697	3142
Mesh	XY	170	10, 17	8	1	9,947058823529412	3633	6627
Mesh	XY	225	15, 15	8	1	10,955555555555556	2991	5757
Circulant	GA	5	1, 2	8	1	1,8	67	131
Circulant	GA	17	1, 4	8	1	2,882352941176470 6	361	674
Circulant	GA	53	1, 13	8	1	5,188679245283019	2081	2350
Circulant	GA	145	1, 36	8	1	10,931034482758621	14764	8159
Circulant	GA	225	1, 56	8	1	15,933333333333334	22233	12632

HDLNoCGen – возможности

Графический интерфейс





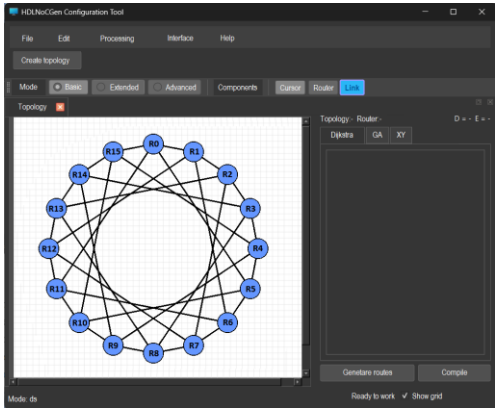


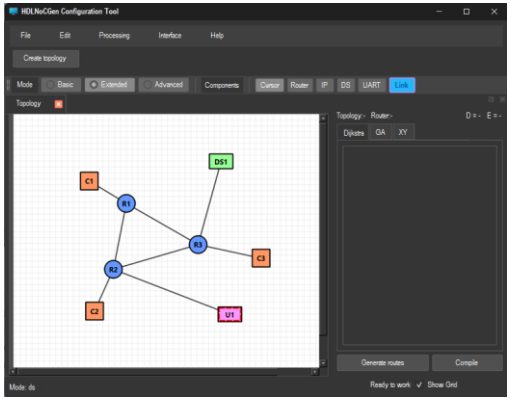
HDLNoCGen – возможности

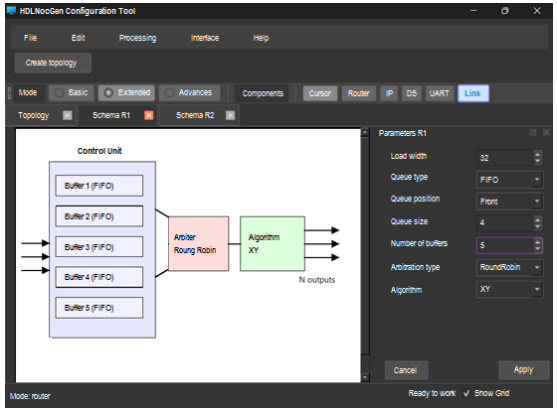
Обновление графического интерфейса

Поддерживает 3 стадии проектирования:

- Математическая модель – структура графа подсистемы связи и анализ алгоритма маршрутизации;
- Базовое представление – отображение всех компонентов СтнК и их параметризация;
- Расширенное представление – демонстрация всех компонентов и связей между ними.



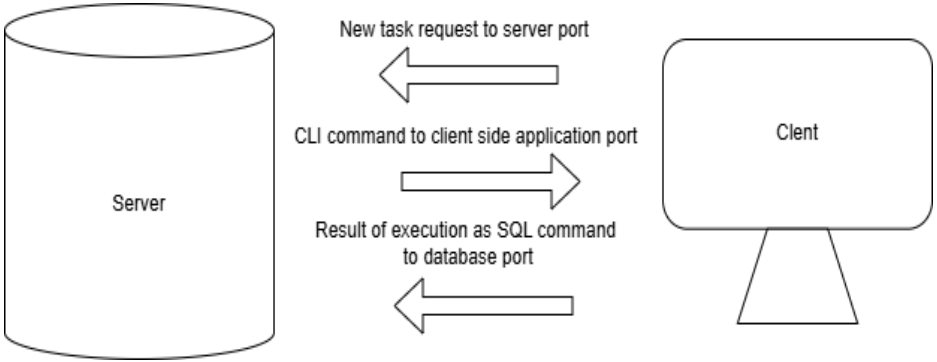




HDLNoCGen – возможности

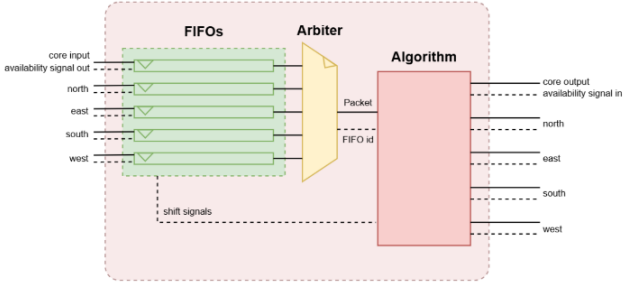
Распределенное моделирование

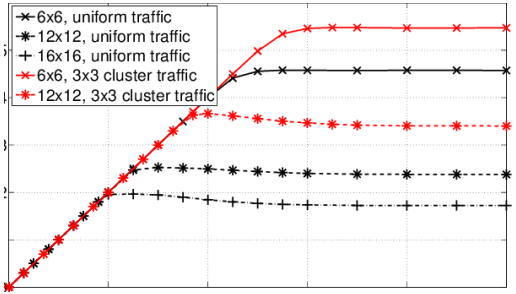
- Клиент-серверная архитектура между массивом компьютеров (клиенты) и SBC Orange-Pi (сервер);
- Сервер по запросу клиента отдает команду для запуска HDLNoCGen.exe с соответствующими параметрами;
- Сервер хостит СУБД PostgreSQL, который указана в приложении HDLNoCGen как целевая для записи данных



Задачи исследований

- Исполняемые в рамках ВКР:
  - Проектирование параметризуемого маршрутизатора с адаптивной длиной пакета;
  - Измерение пропускной способности СтнК;
  - Рекомендательная система для проектирования СтнК;
  - Разработка программно-аппаратного комплекса для прототипирования СтнК для Xilinx, Gowin;
- Текущие подзадачи:
  - Использование существующих стандартов передачи данных;
  - Разработка СтнК с передачей пакетов сегментами.





Структура СтнК

Составляющие модули маршрутизатора:

- N очередей длины M;
- Арбитратор с N входами и 1 выходом;
- Алгоритм с 1 входом и N выходами;

Варианты соединения:

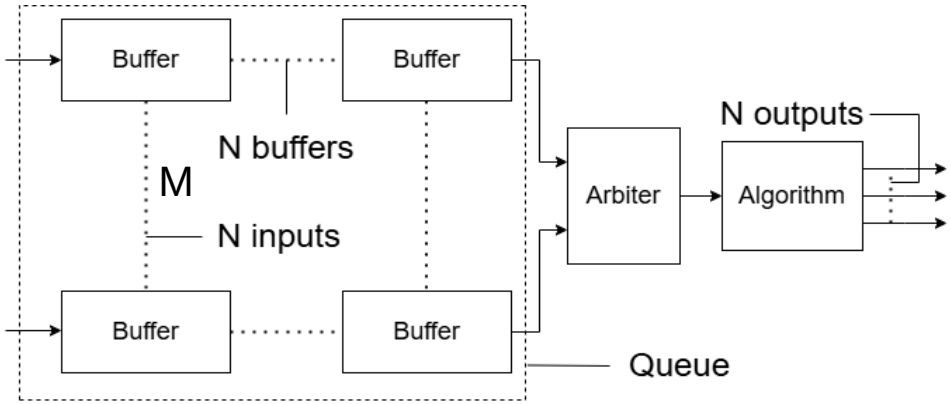
- Очереди-арбитратор-алгоритм;
- Арбитратор-алгоритм-очереди.

Процессорные ядра:

- schoolRISCV;
- picorv32;

Коммуникация между маршрутизатором и процессорным ядром:

- Интерфейс AXI

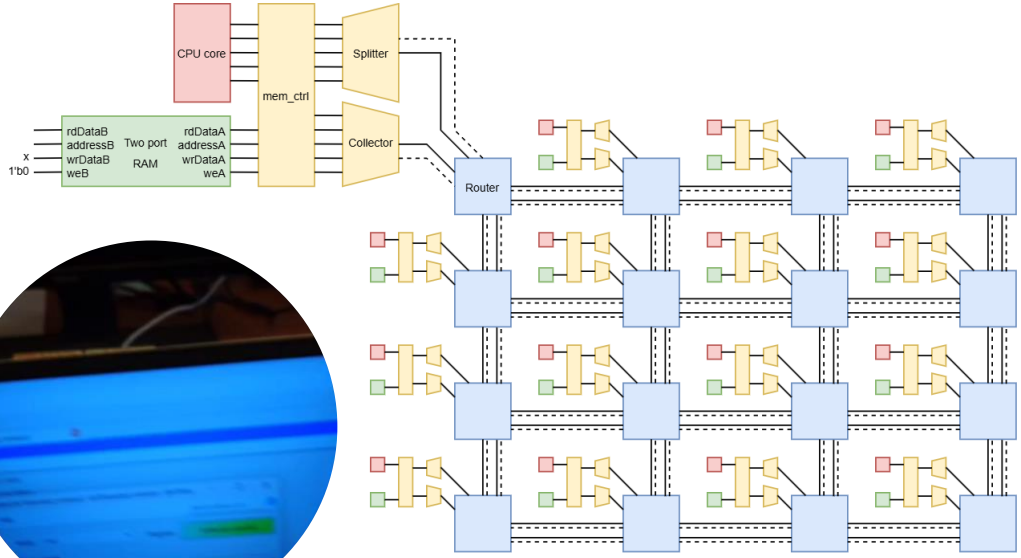


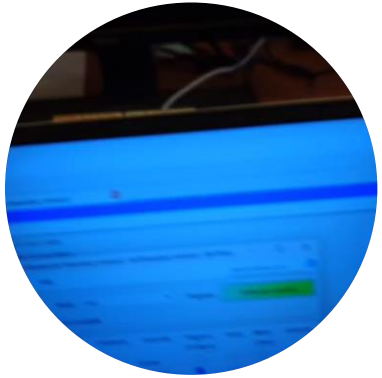
Практическое применение СтнК

Разработка СтнК для проектирования ЦПУ

Первый прототип – процессор на СтнК с делителями и сборщиками пакетов на разделяемой памяти.

- К СтнК были подключены ядра SchoolRISCV и слайсы памяти;
- Модули, использованные для подключения:
    - Контроллеры СтнК;
    - Делители и сборщики пакетов;
- Круговая арбитрация каналов в маршрутизаторах.





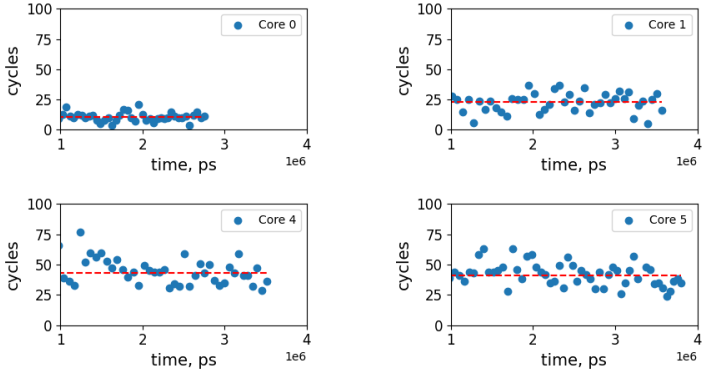
Практическое применение СтнК

Сегментация передачи пакетов

Преимущества:
- Маршрутизаторы остались неизменны;

Недостатки:
- Нельзя гарантировать целостность передачи;
- Низкая производительность;
- Огромные ресурсы, занимаемые сборщиками.

Entity	Average ALUTs	Average registers	Average DSP blocks
schoolRISCV core	1372.9	1032.0	2
memory controller	116.0	211.0	0
splitter	27.3	112.0	0
collector	659.5	416.0	0
central router	516.3	710.0	0
edge router	431.1	560.0	0
corner router	307.0	408.5	0

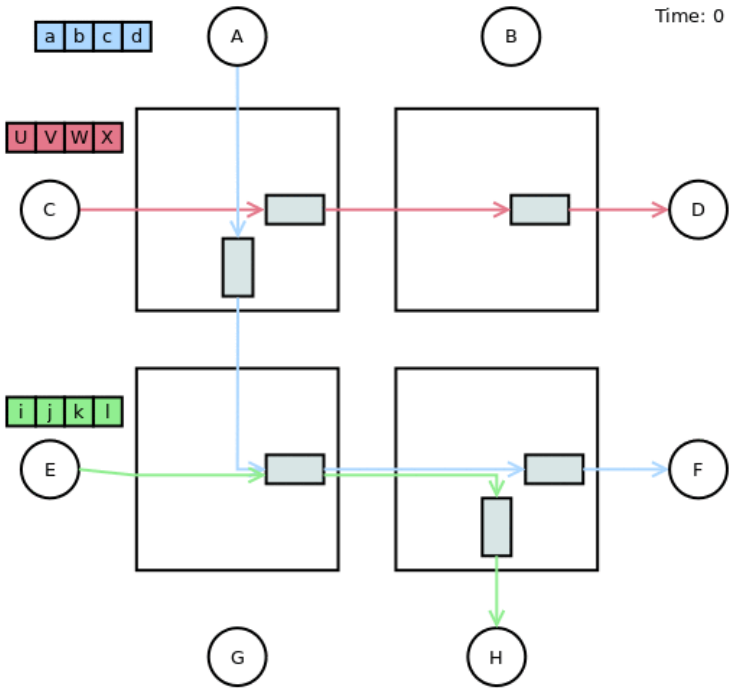


Структура СтнК

Cut-through маршрутизация

Принцип работы:
- Пакет по сети транспортируется сегментами;
- Пакет проходит непрерывным паровозом – «локомотивом» выступает первый сегмент с информацией для маршрутизации, а между «вагонами» никакие части чужих пакетов не просачиваются;

Преимущества:
- Более высокая плотность передачи пакетов различной длины;
- Уменьшенная задержка передачи пакетов;
- Гибкость выбора ширины канала;

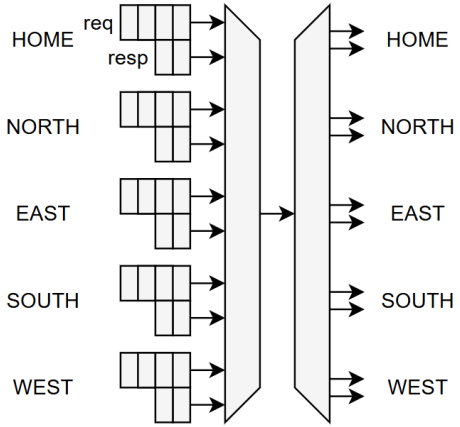


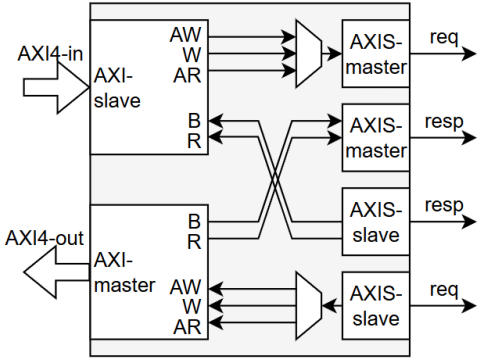
Практическое применение СтнК

Разработка СтнК для свертки изображений

Текущая реализация: XY Mesh 3x3

- Устройства подключаются к СтнК через AXI4-MM;
- Внутри интерконнекта передача данных идет через AXI4-Stream;
- Между каждой парой master-slave устройств и СтнК установлен четырехканальный двунаправленный AXI4-to-AXI4S бридж;
- Арбитр и алгоритм следят за целостностью передачи всех цепочек;
- Каналы реквестов и респонсов разделены.

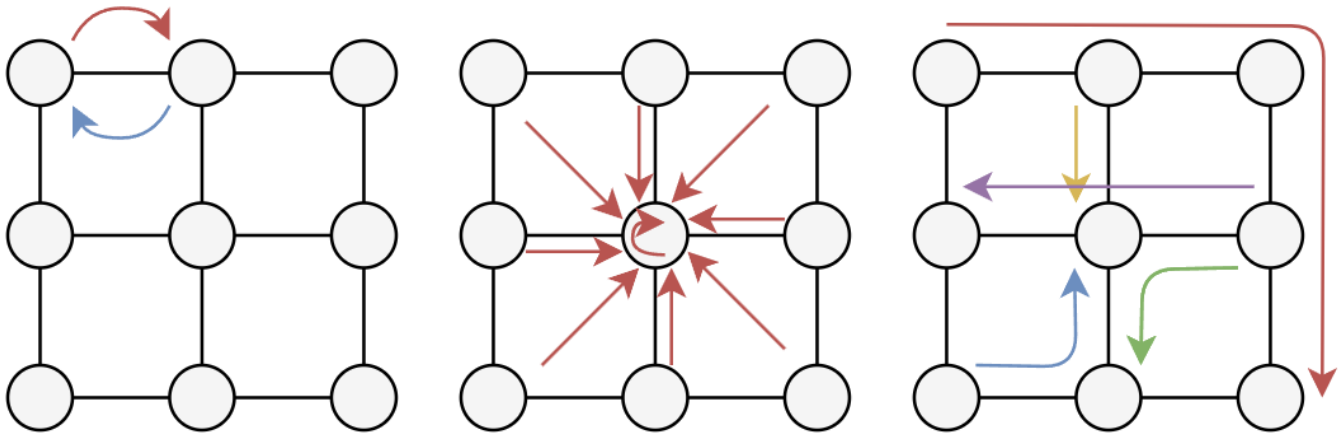




Практическое применение СтнК

Предварительные тесты

Виды трафика:
- Диалог;
- Все в одного;
- Множество прогонов каждого в случайного.



Практическое применение СтнК

Система сборки

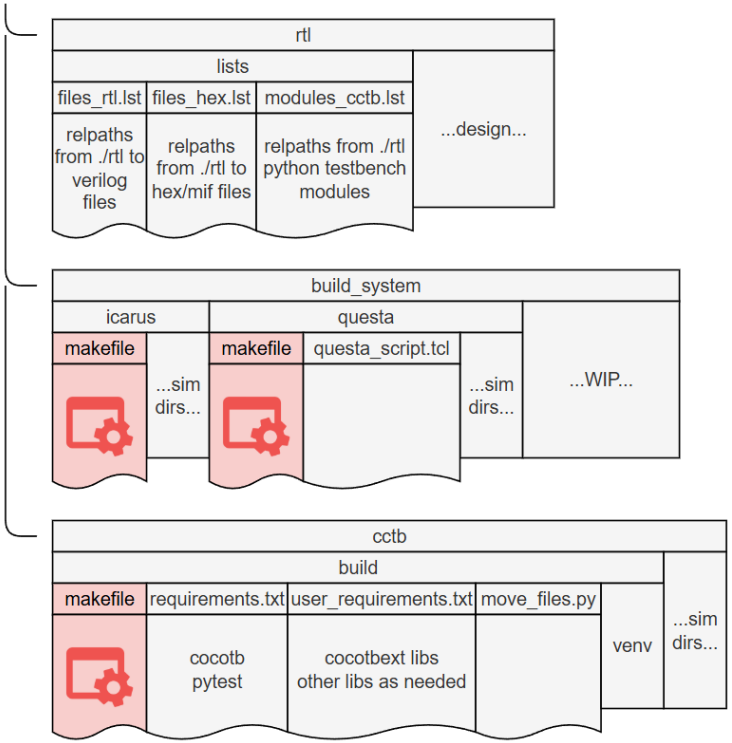
Система сборки построена по принципу плоской сборки, что позволяет избавиться от `include в коде дизайна.

Ключевые особенности:

- Входной точкой каждого симулятора является соответствующий makefile;
- Проверена поддержка icarus и questa для классических .sv тестбенчей, проверена работа questa через cocotb для python тестбенчей;

Особенности cocotb:

- При первом запуске ставится venv в директорию cctb/build. Далее он ставится снова только при обновлении .txt файлов.



Практическое применение СтнК

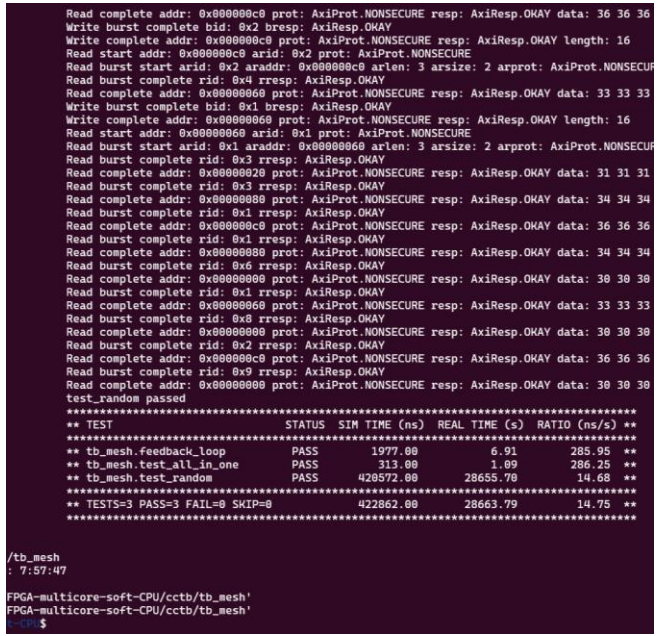
Результаты работы

Характеристики данных:

- Количество данных в транзакции: 128 бит;
- Размер одного трансфера: 32 бит;
- Размер каждого FIFO маршрутизаторов: 2;
- Атомарный элемент тестирования – операция записи, после которой сразу начинается операция чтения того же адреса;

Нагрузка теста:

- Диалог:
  - 50 атомарных операций подряд;
- Все в одного:
  - Каждый одновременно запускает 4 атомарные операции в средний узел;
- Каждый в случайного:
  - Каждый одновременно запускает 4 атомарные операции в случайный узел. 1000 прогонов.



```
Read complete addr: 0x000000c0 prot: AxiProt.NONSECURE resp: AxiResp.OKAY data: 36 36 36
Write burst complete bid: 0x2 bresp: AxiResp.OKAY
Write complete addr: 0x000000c0 prot: AxiProt.NONSECURE resp: AxiResp.OKAY length: 16
Read start addr: 0x000000c0 arid: 0x2 prot: AxiProt.NONSECURE
Read burst start arid: 0x2 araddr: 0x000000c0 arlen: 3 arsize: 2 arprot: AxiProt.NONSECUR
Read burst complete rid: 0x4 rresp: AxiResp.OKAY
Read complete addr: 0x00000060 prot: AxiProt.NONSECURE resp: AxiResp.OKAY data: 33 33 33
Write burst complete bid: 0x1 bresp: AxiResp.OKAY
Write complete addr: 0x00000060 prot: AxiProt.NONSECURE resp: AxiResp.OKAY length: 16
Read start addr: 0x00000060 arid: 0x1 prot: AxiProt.NONSECURE
Read burst start arid: 0x1 araddr: 0x00000060 arlen: 3 arsize: 2 arprot: AxiProt.NONSECUR
Read burst complete rid: 0x3 rresp: AxiResp.OKAY
Read complete addr: 0x00000020 prot: AxiProt.NONSECURE resp: AxiResp.OKAY data: 31 31 31
Read burst complete rid: 0x3 rresp: AxiResp.OKAY
Read complete addr: 0x00000080 prot: AxiProt.NONSECURE resp: AxiResp.OKAY data: 34 34 34
Read burst complete rid: 0x1 rresp: AxiResp.OKAY
Read complete addr: 0x000000c0 prot: AxiProt.NONSECURE resp: AxiResp.OKAY data: 36 36 36
Read burst complete rid: 0x1 rresp: AxiResp.OKAY
Read complete addr: 0x00000080 prot: AxiProt.NONSECURE resp: AxiResp.OKAY data: 34 34 34
Read burst complete rid: 0x6 rresp: AxiResp.OKAY
Read complete addr: 0x00000000 prot: AxiProt.NONSECURE resp: AxiResp.OKAY data: 30 30 30
Read burst complete rid: 0x1 rresp: AxiResp.OKAY
Read complete addr: 0x00000060 prot: AxiProt.NONSECURE resp: AxiResp.OKAY data: 33 33 33
Read burst complete rid: 0x8 rresp: AxiResp.OKAY
Read complete addr: 0x00000000 prot: AxiProt.NONSECURE resp: AxiResp.OKAY data: 30 30 30
Read burst complete rid: 0x2 rresp: AxiResp.OKAY
Read complete addr: 0x000000c0 prot: AxiProt.NONSECURE resp: AxiResp.OKAY data: 36 36 36
Read burst complete rid: 0x9 rresp: AxiResp.OKAY
Read complete addr: 0x00000000 prot: AxiProt.NONSECURE resp: AxiResp.OKAY data: 30 30 30
test_random passed
**************************************************************************************
** TEST                          STATUS  SIM TIME (ns)  REAL TIME (s)  RATIO (ns/s) **
**************************************************************************************
** tb_mesh.feedback_loop         PASS        1977.00           6.91        285.95  **
** tb_mesh.test_all_in_one       PASS         313.00           1.09        286.25  **
** tb_mesh.test_random           PASS      420572.00       28655.70         14.68  **
**************************************************************************************
** TESTS=3 PASS=3 FAIL=0 SKIP=0              422862.00       28663.79         14.75  **
**************************************************************************************


/tb_mesh
: 7:57:47

FPGA-multicore-soft-CPU/cctb/tb_mesh'
FPGA-multicore-soft-CPU/cctb/tb_mesh'
-CPU$
```

ВЫСШАЯ • ШКОЛА • ЭКОНОМИКИ •

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ