МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
(государственный университет)
Факультет радиотехники и кибернетики
Кафедра информатики и вычислительной техники

Разработка контроллера когерентности памяти для МП «МЦСТ-R2000»

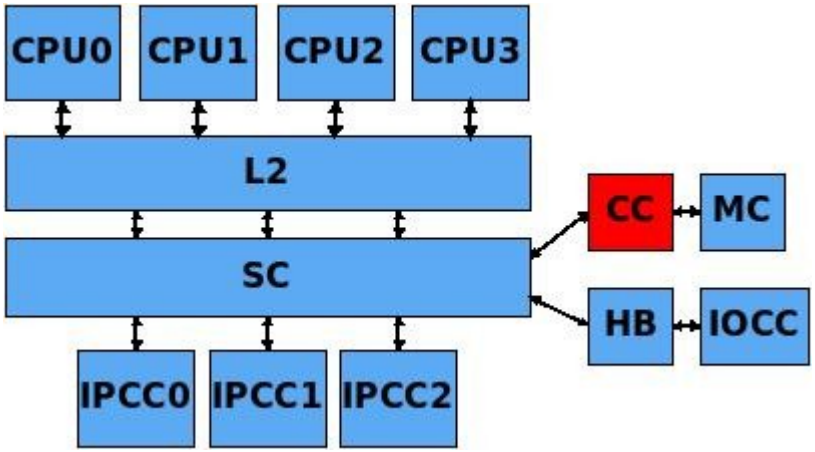
Выпускная квалификационная работа магистра
(Магистерская диссертация)

Научное руководство: Черепанов С.А.
к.т.н. Ганжа Т.В.

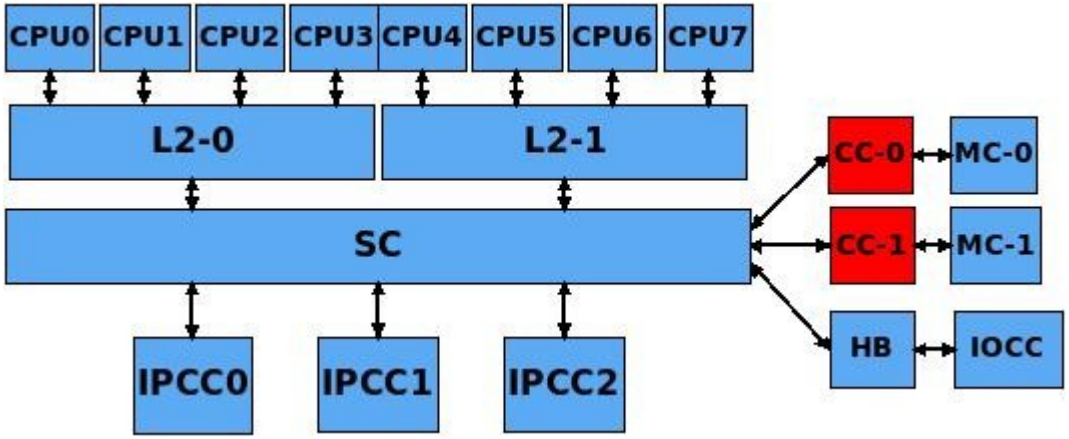
Студент: Грачик В.И, ФРТК 113 группа

Мотивация

R1000:



R2000:



	R1000	R2000
Одновременно обрабатываемых транзакций	16	32
Разрядность шины данных с комутатором (SC) и контроллером памяти (MC)	64/128	128/256
Поддерживаемая оперативная память	DDR2	DDR4
Контроллеров памяти	1	2

Цель работы

Разработка контроллера когерентности памяти (CC) для микропроцессора МЦСТ-R2000

Требуемая функциональность:

- Отправка снуп-запросов всем кэшам L2 и сбор ответов
- Сериализация запросов в MC
- Отслеживание и разрешение конфликтов по адресам
- Поддержка интерливинга
- Хранение состояний кэш-строк, используемых во всех кэшах L2 (Кэш-директория)
- Поддержка ECC в DRAM и кэш-директории

За основу был взят контроллер из микропроцессора МЦСТ-R1000

Задачи

- Отправка снуп-запросов всем кэшам L2 и сбор ответов
  - ✔ Основной функционал реализован в R1000
  - ✔ Модифицированна схема вычисления маски для CWM-запросов
  - ✔ Увеличена разрядность шины данных с SC

- Сериализация запросов в MC
  - ✔ Переделан интерфейс с MC
  - ✔ Увеличена разрядность шины данных с MC

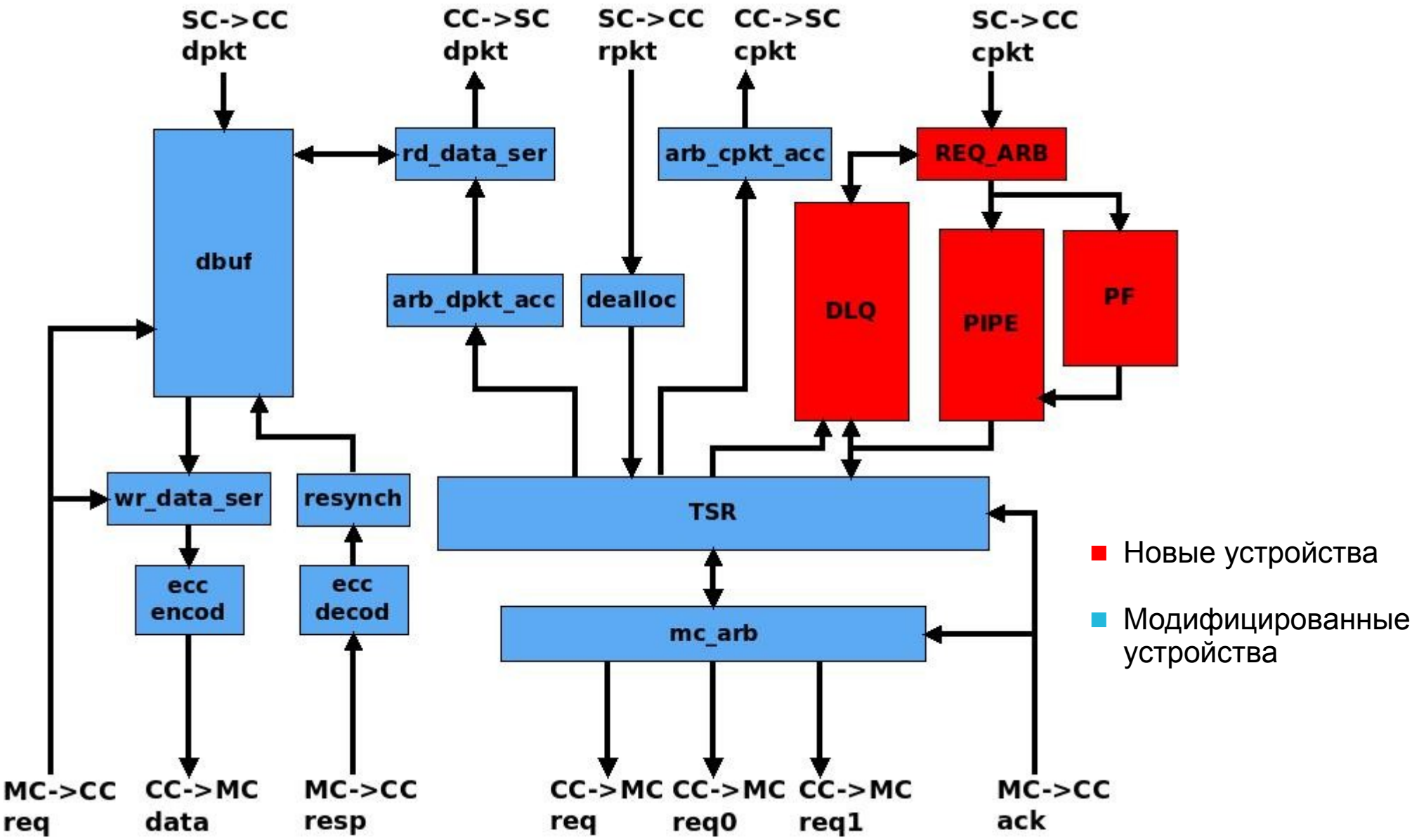
- Отслеживание и разрешение конфликтов по адресам и сохранение порядка выполнения DMA-запросов
  - ✔ Реализован адресный конвейер CC
  - ✔ Реализован буфер отложенных запросов (DLQ)

- Поддержка интерливинга
  - ✔ Добавлена логика исключения бита интерливинга на входе в CC и включения бита интерливинга в физические адреса снуп-запросов

- Хранение состояний кэш-строк, используемых во всех кэшах
  - ✔ Реализована кэш-директория (PF)

- Поддержка ECC в DRAM и кэш-директории
  - ✔ Поддержка ECC в DRAM реализована в R1000
  - ✔ Добавлена поддержка ECC в кэш-директории

Общая схема CC



Общая схема CC

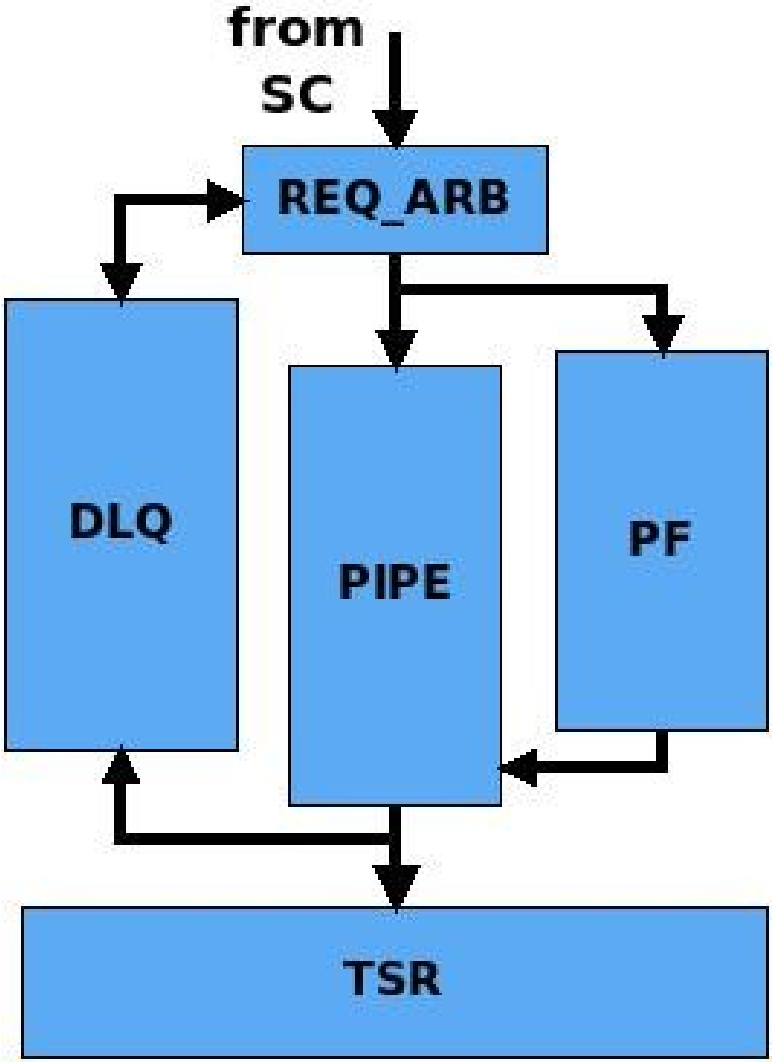
REQ_ARB – арбитр запросов.
Осуществляет арбитраж запросов, пришедших от системного коммутатора и из буфера отложенных запросов.

DLQ – буфер отложенных запросов.
Содержит запросы, получившие блокировку при прохождении конвейера.

PF – кэш-директория.

PIPE – адресный конвейер CC. Разрешает адресные коллизии, содержит запросы, находящиеся на обработке в PF.

TSR – буфер запросов, находящихся на исполнении.



CC: Подзадачи

- Реализовать в контроллере когерентности кэш-директорию(PF), хранящую информацию о всех кэш-строках памяти процессора, содержащихся во всех кэш-памятях системы.

- Разработать конвейер CC с учётом новых блокировок, которые появляются после введения кэш-директории.

- Реализовать буфер отложенных запросов (DLQ), который бы хранил входящие запросы, получившие блокировку при прохождении конвейера.

Кэш-директория: общие принципы

- Используется кэш-память с ассоциативностью 16
- Протокол поддержки когерентности: MOSI
- Ячейки директории имеют следующий формат:

state [4:3] – идентификатор узла
state [2] – номер L2-кэша
state [1:0] – код состояния

- Владелец строки хранится только в состояниях M и O. Для состояния S используются широковещательные снуп-запросы.

- Состояние ячейки изменяется при получении запросов:
  - **CRD** – когерентное чтение
  - **CRI** – когерентное чтение с инвалидированием
  - **CI** – когерентное инвалидирование
  - **CRS** – когерентное чтение блока без изменения состояния
  - **CWB** – когерентный Write-Back модифицированных данных
  - **CWI** – когерентная запись с инвалидированием
  - **CWM** – когерентная запись с маской

- Особое событие, генерируемое кэш-директорией:
  - **CWD** – вытеснение строки из кэш-директории

Кэш-директория: поддерживаемый протокол когерентности

Переходы между состояниями:

	I	O	S	M
CRD	M	O	S	O
CRI	M	M	M	M
CI	M	M	M	M
CRS	I	O	S	M

	I	O	S	M
CWB	I	S/O	S	I
CWI	I	I	I	I
CWM	I	I	I	I

Рассылаемые снуп-запросы:

	I	O	S	M
CRD	–	**D**CRD	–	**D**CRD
CRI	–	**B**CRI	**B**CRI	**D**CRI
CI	–	**B**CRI	**B**CRI	**D**CRI
CRS	–	**D**CRD_nc	–	**D**CRD_nc

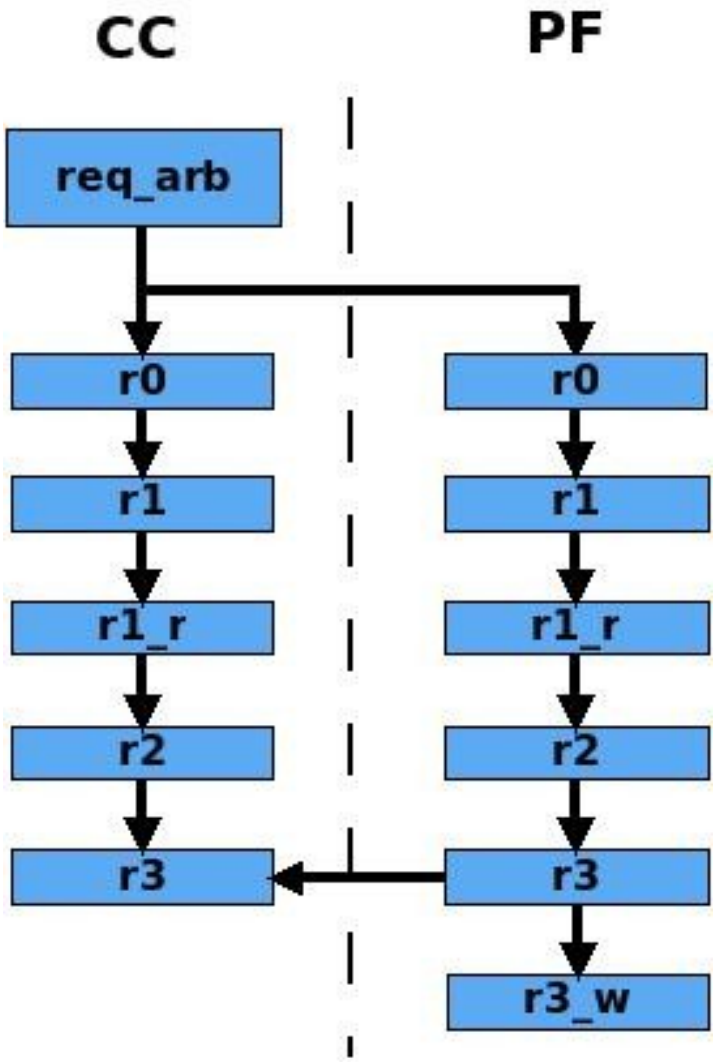
	I	O	S	M
CWB	–	–	–	–
CWI	–	**B**CI	**B**CI	**D**CI
CWM	–	**B**CRI	**B**CRI	**D**CRI
CWD	–	**B**CRI	**B**CRI	**D**CRI

B - широковещательный снуп-запрос
D – прямой снуп-запрос

Введение кэш-директории позволяет:

- ✔ Сократить время доступа в незакэшированную память
- ✔ Уменьшить объём когерентного трафика

Кэш-директория: конвейер



a: Получение индекса* для обращения к кэш-директории из адреса запроса

r0-r1: Чтение памяти директории

r1_r: 1) Декодирование ECC
2) Поиск ячейки

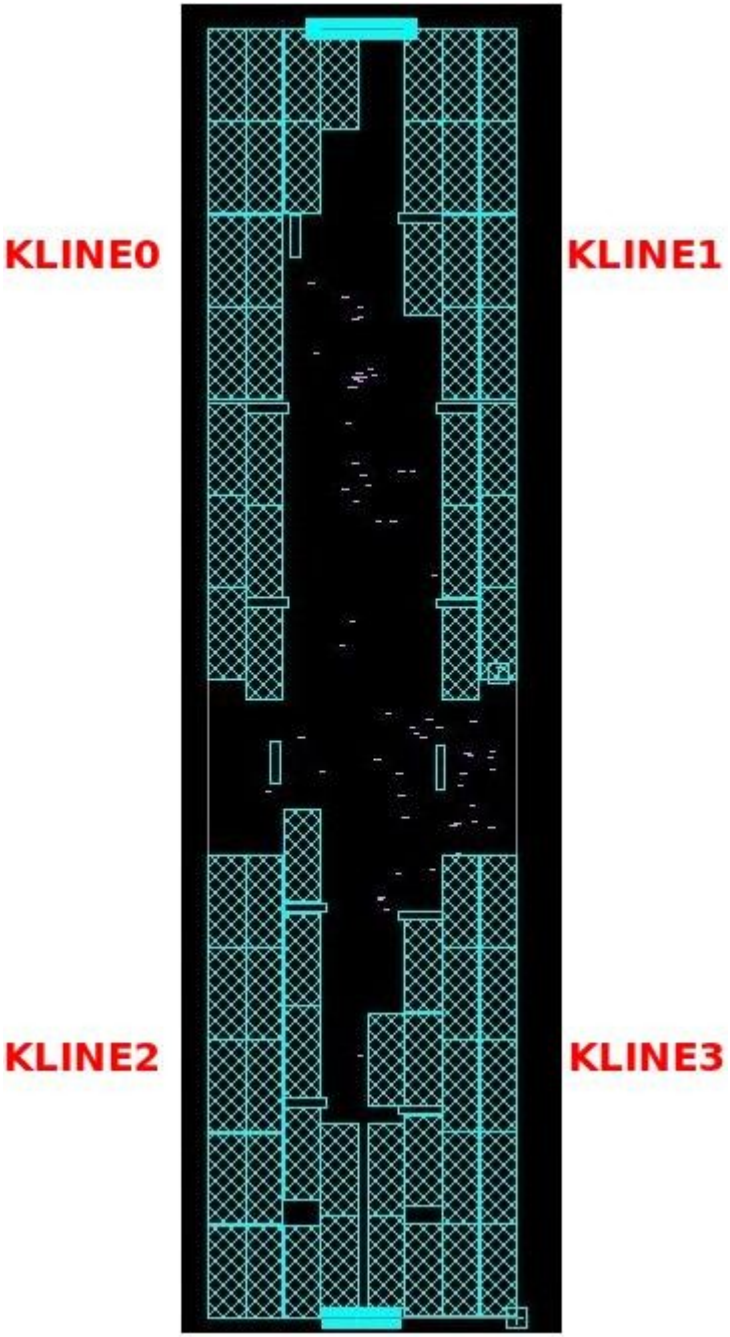
r2: 1) Чтение ячейки
2) Формирование нового состояния

r3: 1) Формирование ECC
2) Выдача атрибутов в конвейер CC

r3_w: Запись ячейки

*получение индекса из адреса запроса:

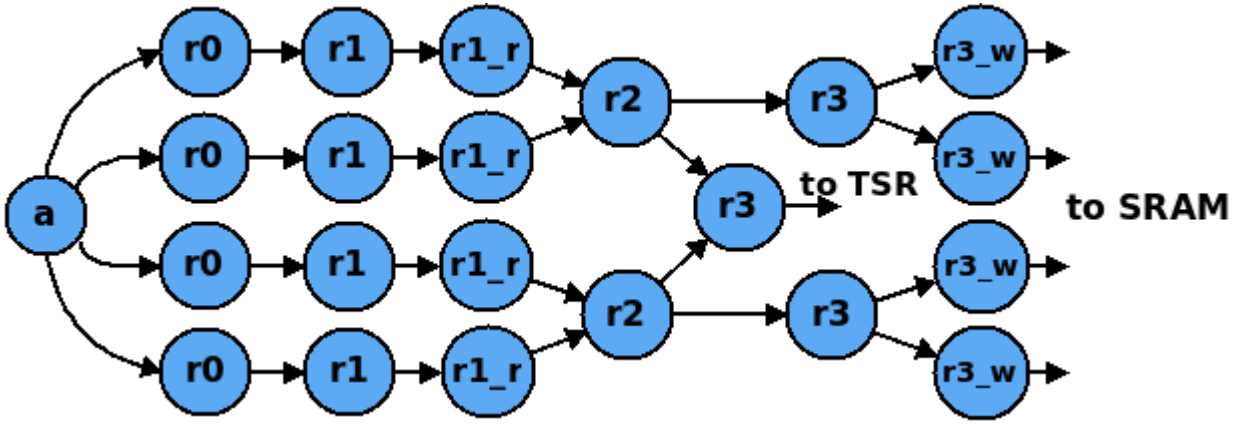
Кэш-директория: организация памяти



Память кэш-директории состоит из 68 памятей типа SRAM размером 1024x32, логически она организована как 4-е строки (kline) по 17 SRAM в каждой.

Из-за проблем с физическим дизайном, в конвейер директории был внесён ряд изменений:

- Фазы r0 – r1_r и r3_w локализованы для каждой kline
- Фазы r2 – r3 локализованы для верхней и нижней части кристала



r1:

r1_r:

r2:

Кэш-директория: параметры

Тип памяти	Множественно ассоциативная память
Объём	256 КБ (с учётом ECC-битов)
Общее количество банок памяти	64 (4 kline (строк банок) по 16 банок)
Размеры одной банки памяти	1024x32
Ассоциативность	16
Размер индекса	12
Размер тэга	21
Длина строки памяти (состояние которой хранится в кэш-директории)	64 байта (одна кэш-строка)
Алгоритм замещения	псевдослучайный ($x^{17} + x^{14} + 1$)
Темп поступления запросов	1 запрос в такт
Время обработки запросов	5 тактов (6 с записью в память)
Размер элемента	32 бита
Количество ECC-бит в строке	6
Количество бит-состояний строки	5

Кэш-директория: влияние на адресный конвейер

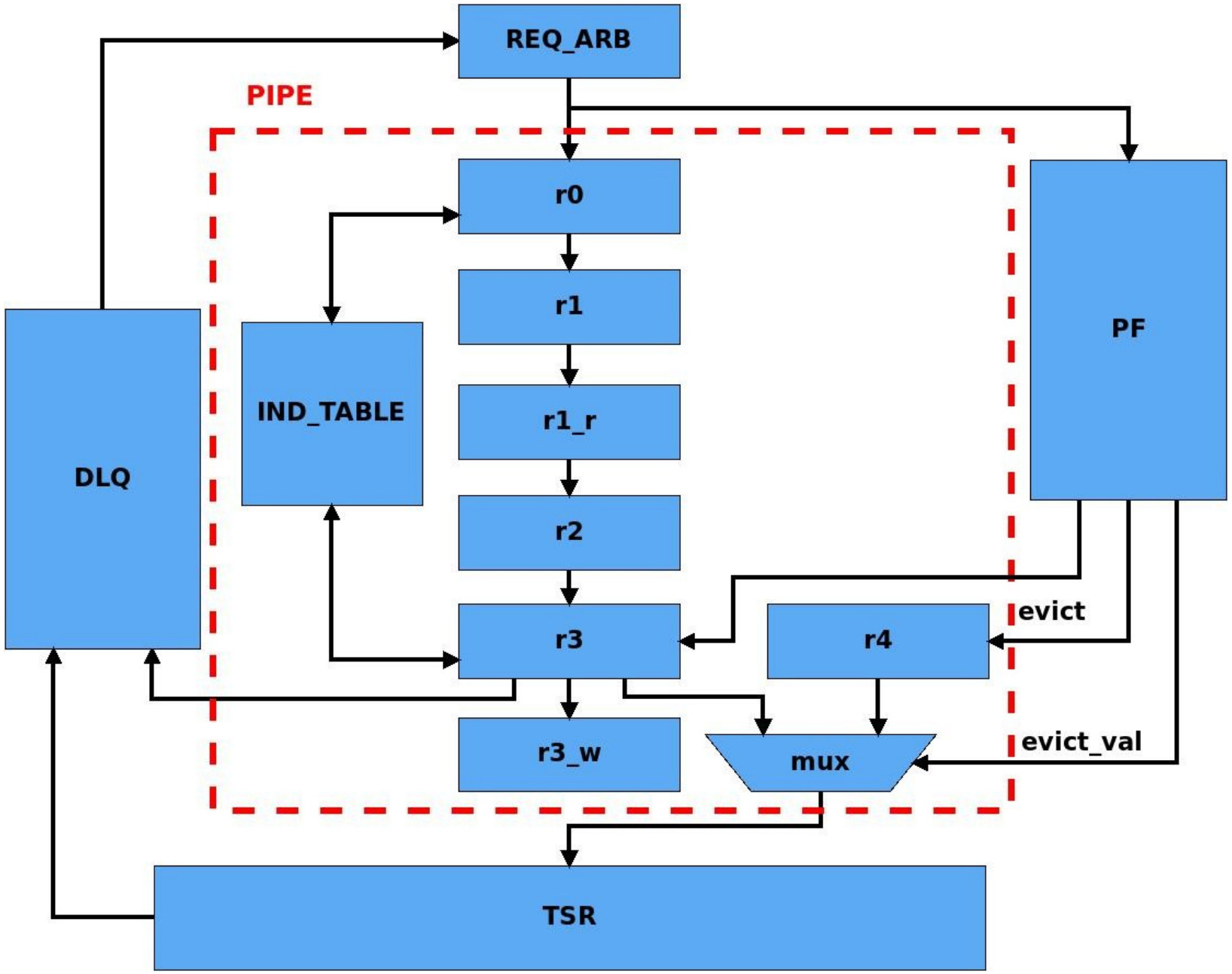
- В конвейере кэш-директории не должно быть запросов с одинаковыми индексами.
- Адресный конвейер СС должен поддерживать обработку вытеснений строки из кэш-директории.
- Появление новых блокировок в конвейере:

  index_busy – генерируется, если в контроллере уже есть 16 запросов с таким же индексом. В этом случае запрос просто отменяется. Вытеснение не генерируется;

  hit_index – срабатывает**,** если индекс запроса совпал с индексом другого запроса, находящегося в конвейере;

  evict_val – отмена запроса из-за вытеснения. При генерации вытеснения, сначала в TSR записывается запрос с r3 (вызвавший вытеснение), следующий запрос отменяется и вместо него в TSR записывается вытеснение.

Конвейер CC: общая схема



Конвейер CC: принцип работы

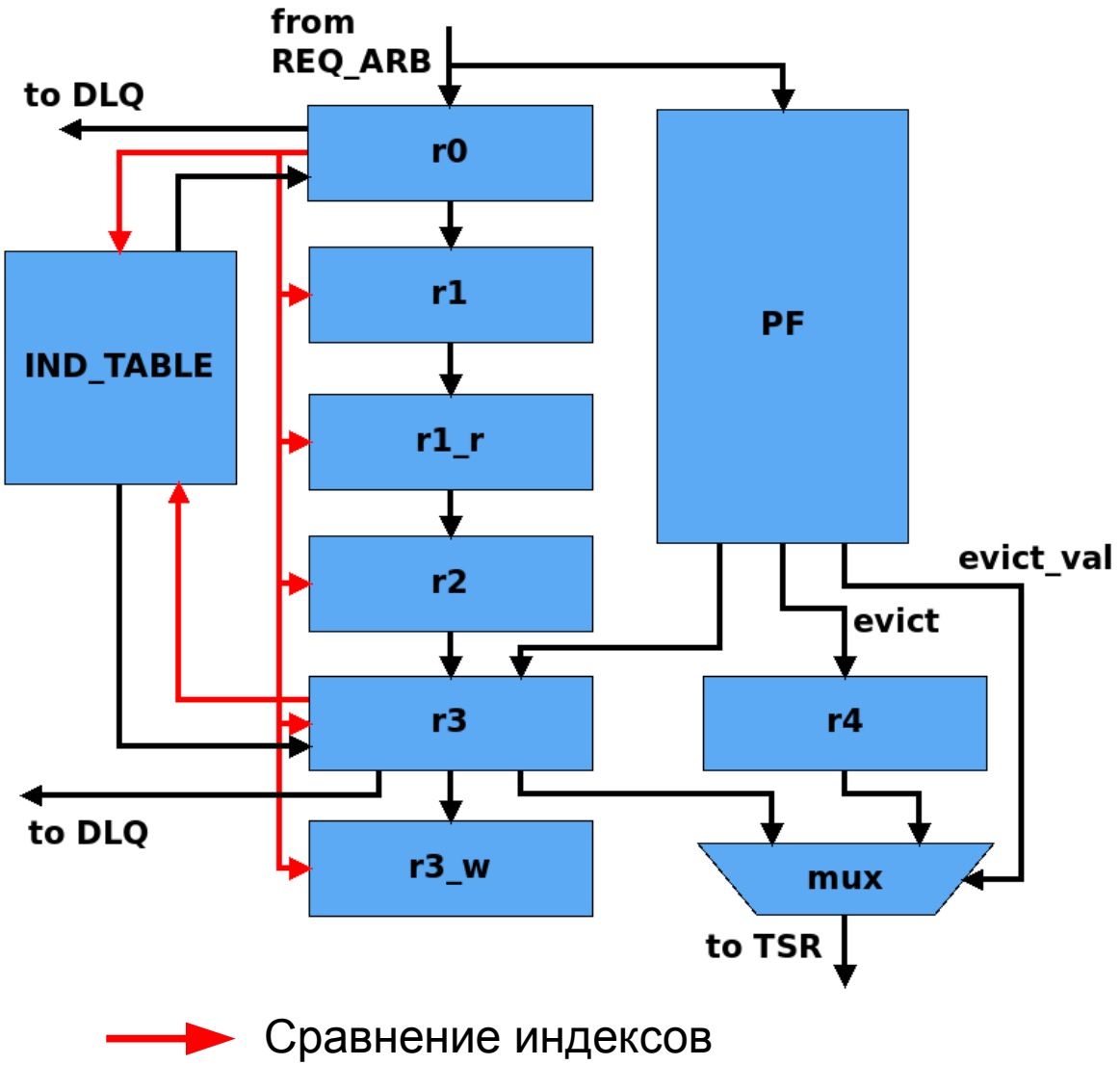
r0: Индекс транзакции сравнивается с индексами транзакций с фаз r1-r3_w и с индексами из IND_TABLE

r1-r1_r: Ожидание ответа от кэш-директории

r2: Сравнение адреса запроса с адресами запросов, находящихся на исполнении (в TSR)

r3: 1) Генерация блокировок
2) Формирование признака записи в DLQ и TSR

r3_w: Содержит индекс запроса, результат обработки которого в данный момент записывается в PF. Используется только для сравнения индексов.



Конвейер CC: IND_TABLE

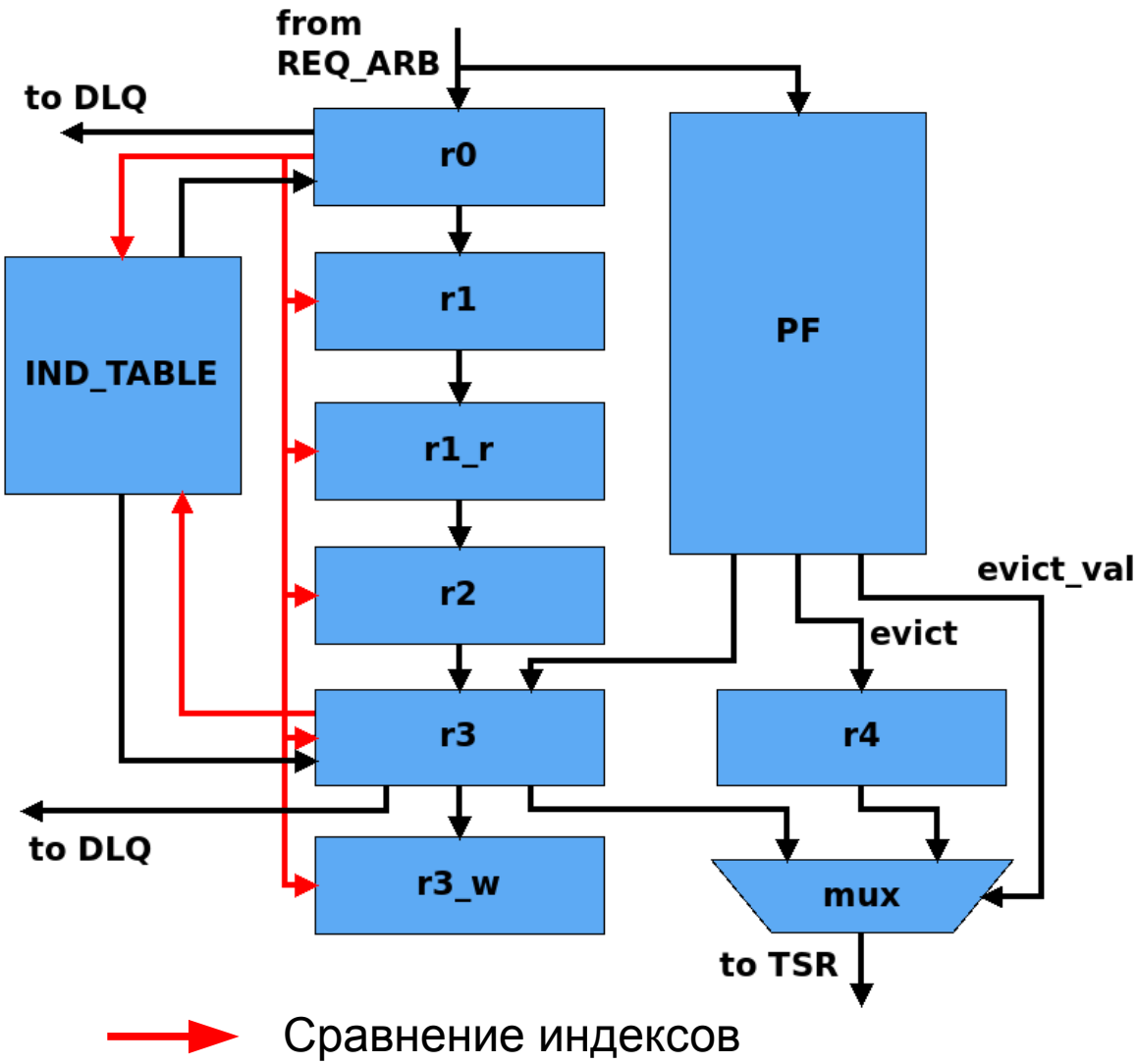
IND_TABLE – структура, хранящая индексы запросов, которые получили блокировку в конвейере CC

- Формат ячеки IND_TABLE:

last_num [5:0]	ind [11:0]

  - **last_num** – номер ячейки в DLQ, в которой находится последняя транзакция с индексом **ind**
  - **ind –** индекс транзакции

- При совпадении индекса транзакции с индексом другой транзакции, вторая транзакция отменяется, а её индекс и номер ячейки DLQ записываются в IND_TABLE.



Конвейер CC: генерируемые блокировки

- Возможные блокировки в конвейере:

rand_blk*

index_busy – занятость сэта в кэш-директории
hit_index – совпадение с индексом запроса, находящегося в конвейере
evict_val – генерация вытеснения на r3

tsr_busy – занятость TSR
dma_blk – выполнение DMA-запросов приостановлено
hit_tsr – совпадение адреса запроса с адресом транзакции обрабатывающейся в TSR

- Возможные блокировки в арбитре:

lose_arb – запрос от коммутатора проиграл арбитраж
dma_blk – выполнение DMA-запросов приостановлено

*__rand_blk__ – случайные блокировки

Буфер отложенных запросов

Назначение:

- Хранение запросов, получивших блокировку при прохождении конвейера

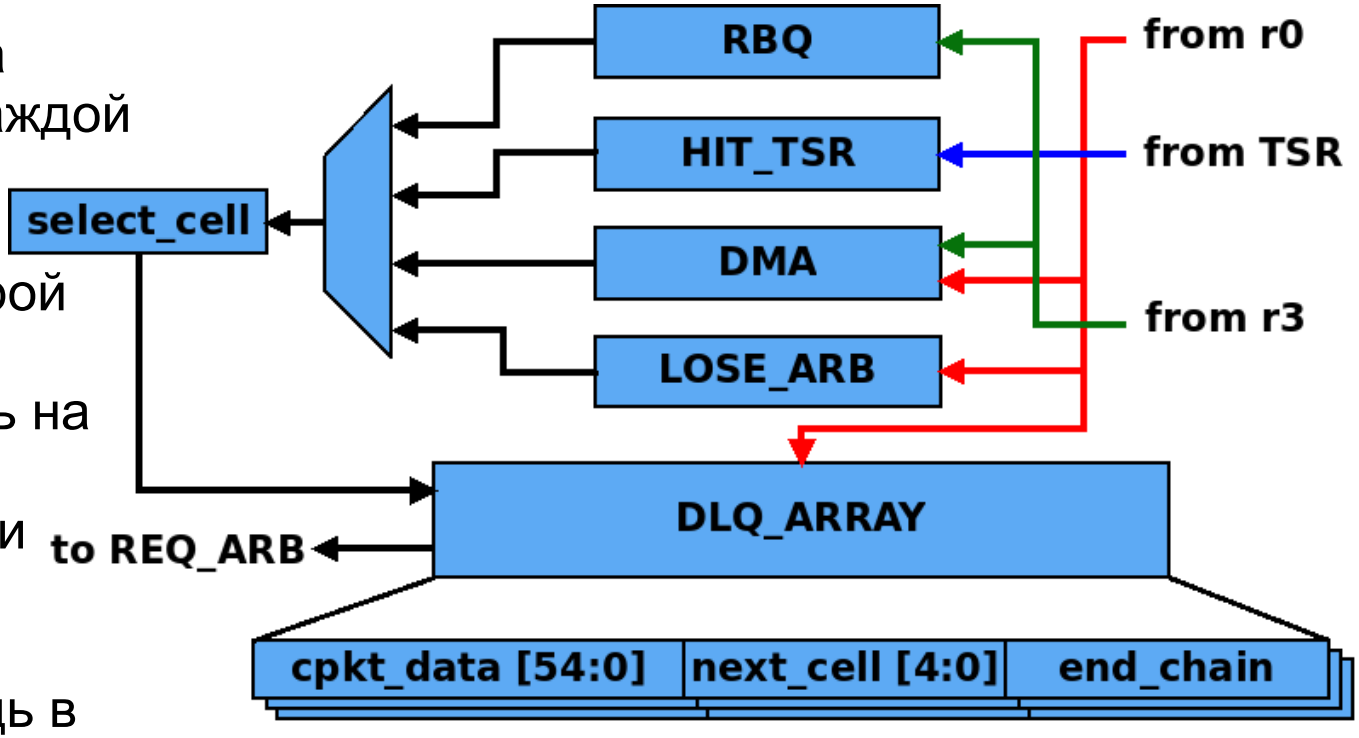
Требования:

- Формирование цепочек запросов с одинаковыми адресами для минимализации адресных сравнений
- Сохранение порядка выполнения DMA-запросов путём формирования цепочек

Буфер отложенных запросов: общая схема

DLQ_ARRAY – память на триггерах, которая для каждой транзакции хранит:

- данные (cpkt_data)
- номер ячейки, в которой лежит следующая транзакция (указатель на транзакцию)
- признак конца цепочки



Каждой блокировке соответствут своя очередь в DLQ:

- **RBQ** – rand_blk
- **HIT_TSR** – hit_tsr
- **DMA** – dma_blk
- **LOSE_ARB** – lose_arb

RBQ, HIT_TSR, DMA, LOSE_ARB – fifo, содержащие указатели на транзакции, записанныее в DLQ_ARRAY

Запись из конвейера CC, на фазе r3

Запись из конвейера CC, на фазе r0 (фактически запись из арбитра)

Запись запроса, вызванного из TSR

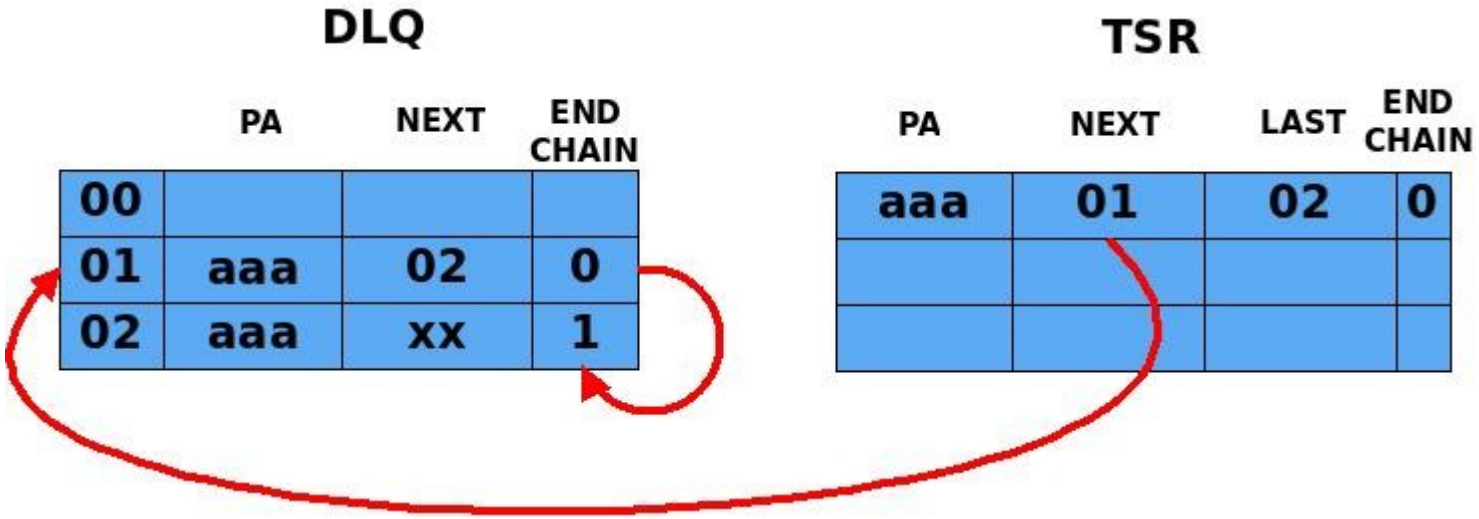
Буфер отложенных запросов: цепочки с одинаковыми адресами

- Формат ячейки TSR:

cpkt_data [55:0]	next [4:0]	last [4:0]	end_chain

  - **cpkt_data –** атрибуты транзакции
  - **next –** указатель на следующую транзакция с таким же адресом
  - **last –** указатель на последнюю транзакцию с таким же адресом
  - **end_chain –** признак конца цепочки из запросов с одинаковыми адресами

- Пример сформированной цепочки:



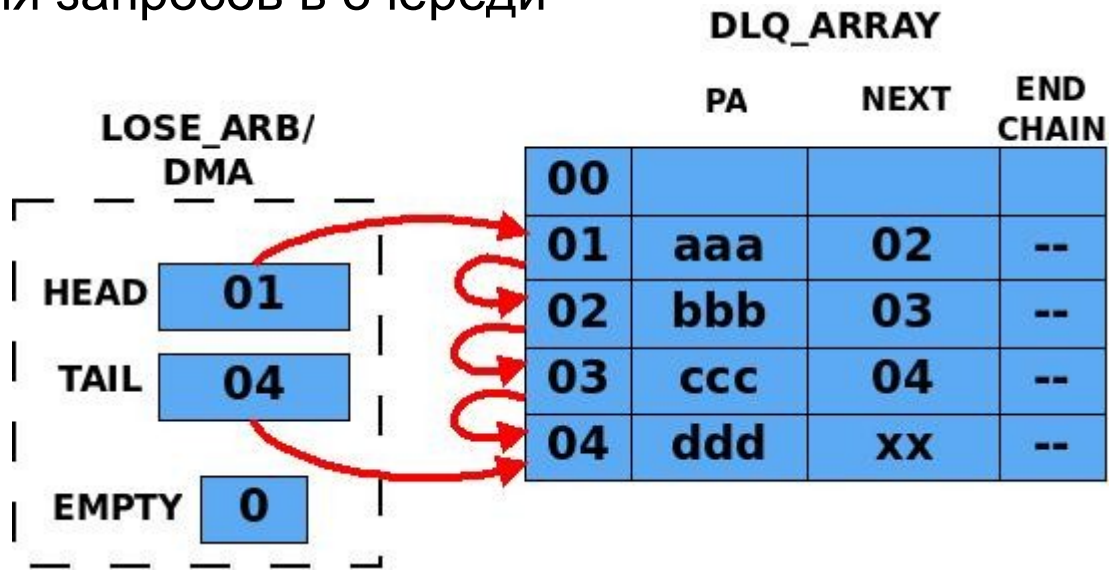
Буфер отложенных запросов: цепочки dma-запросов

- Для сохранения порядка выполнения DMA-запросов в DLQ формируются цепочки DMA-запросов.

- Организация очередей DMA и LOSE_ARB:
  - **HEAD** – указатель на первый запрос в очереди
  - **TAIL** – указатель на последний запрос в очереди
  - **EMPTY** – признак наличия запросов в очереди

- HEAD записывается при записи первого запроса в очередь. При выдачи запроса из очереди в HEAD записывается поле NEXT, из ячейки, на которую указывает HEAD.

- TAIL записывается при добавлении новой транзакции в очередь.



Буфер отложенных запросов: параметры

Размер DLQ	32 ячейки
Размер элемента DLQ	55 бит + 5 + 2
Размер указателя на ячейку в DLQ	5 бит
Время прохождения DLQ	2 такта
Размер RBQ	8 ячеек
Размер HIT_TSR	10 ячеек
Размер DMA/LOSE	2 ячейки
Размер IND_TABLE	7 ячеек

Результаты работы

- Разработан контроллер когерентности памяти для микропроцессора МЦСТ-R2000

- Проведено автономное тестирование
  (65 тестов на конвейер CC /IND_TABLE/DLQ,
  30 на кэш-директорию)

- Устройства отлажены на общесистемных тестах

- RTL оптимизирован согласно требованиям физического дизайна:
  - Целевая частота: 2GHz
  - Занимаемая площадь: ~2,1 мм^2

Спасибо за внимание!

Буфер отложенных запросов: принцип работы

rO – получение транзакции от арбитра и атрибутов, которые определяют, куда будет произведена запись:

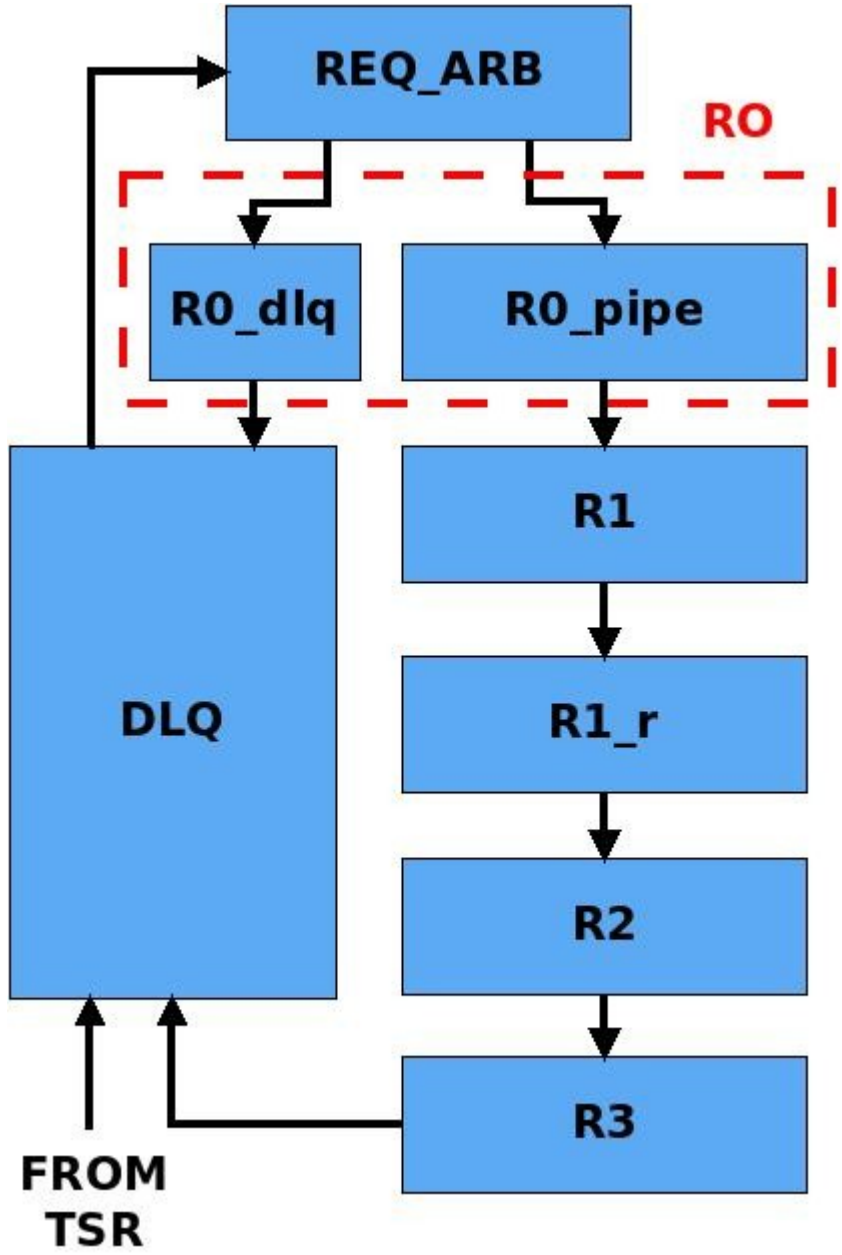
- **to_lose** – запись в очередь LOSE_ARB
- **to_dma** – запись в очередь DMA

Если не ни один из атрибутов не выставлен, то запись будет выполнена только в память DLQ_ARRAY, а очередь будет определена после прохождения конвейера

r1-r2 – ожидание ответа от конвейера

r3 – получение указателя на транзакцию и результатов прохождения конвейера:

- **write_cancel –** конвейер пройден без блокировок. При этом будет освобождена соответствующая ячейка в DLQ.
- **write_confirm –** в процессе прохождения конвейера была сгенерирована блокировка. Будет осуществлена запись в одну из очередей.
- **to_tsr_req/to_dma/to_rbq –** указание очереди, в которую будет записан запрос

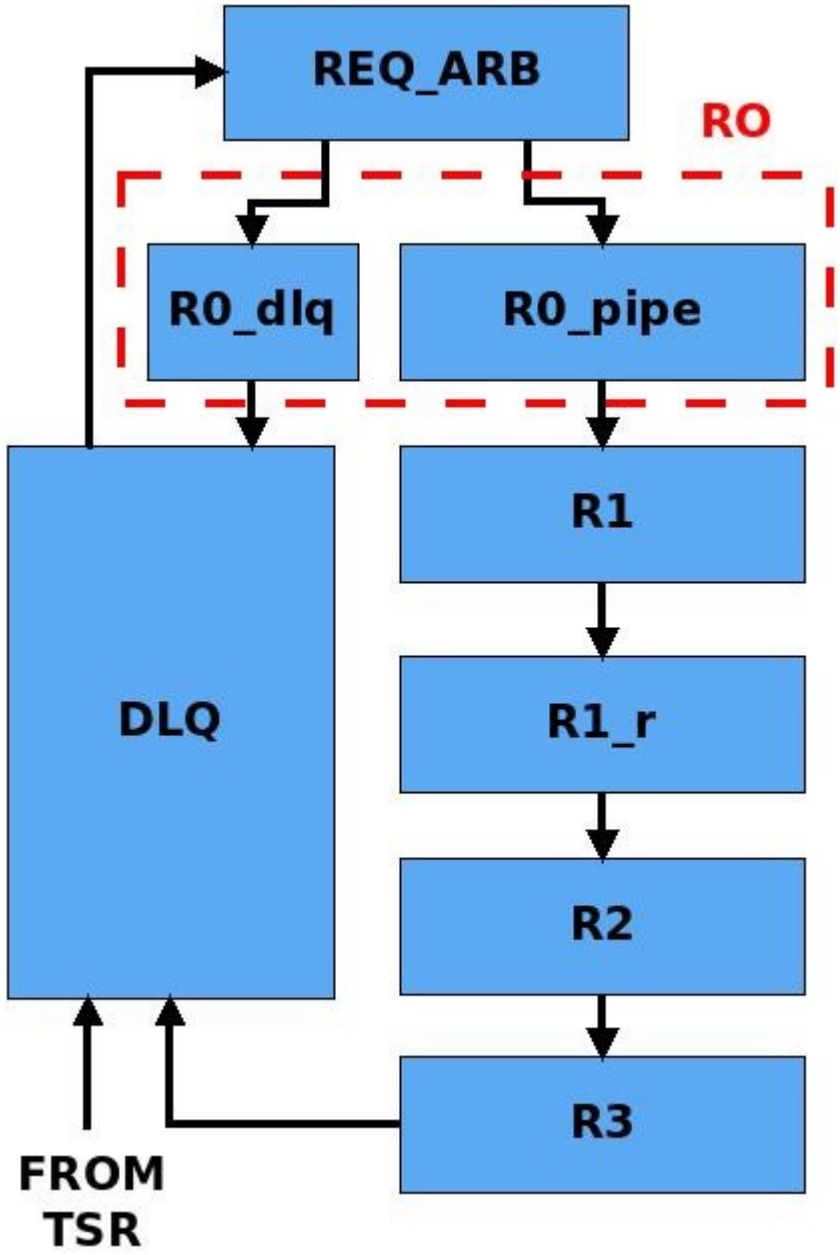


Буфер отложенных запросов: принцип работы

- Если при прохождении конвейера было сгенерировано сразу несколько блокировок, обрабатывается та, у которой больший приоритет:

Приоритет	Блокировка
0	dma_blk
1	hit_tsr
2	rand_blk

- Асинхронно с выдачей указателя на транзакцию и блокировки на r3, в DLQ может прийти запрос из TSR на выдачу следующей транзакции. (Происходит только в том случае, если раньше была сформирована цепочка из запросов с одинаковыми адресами.)

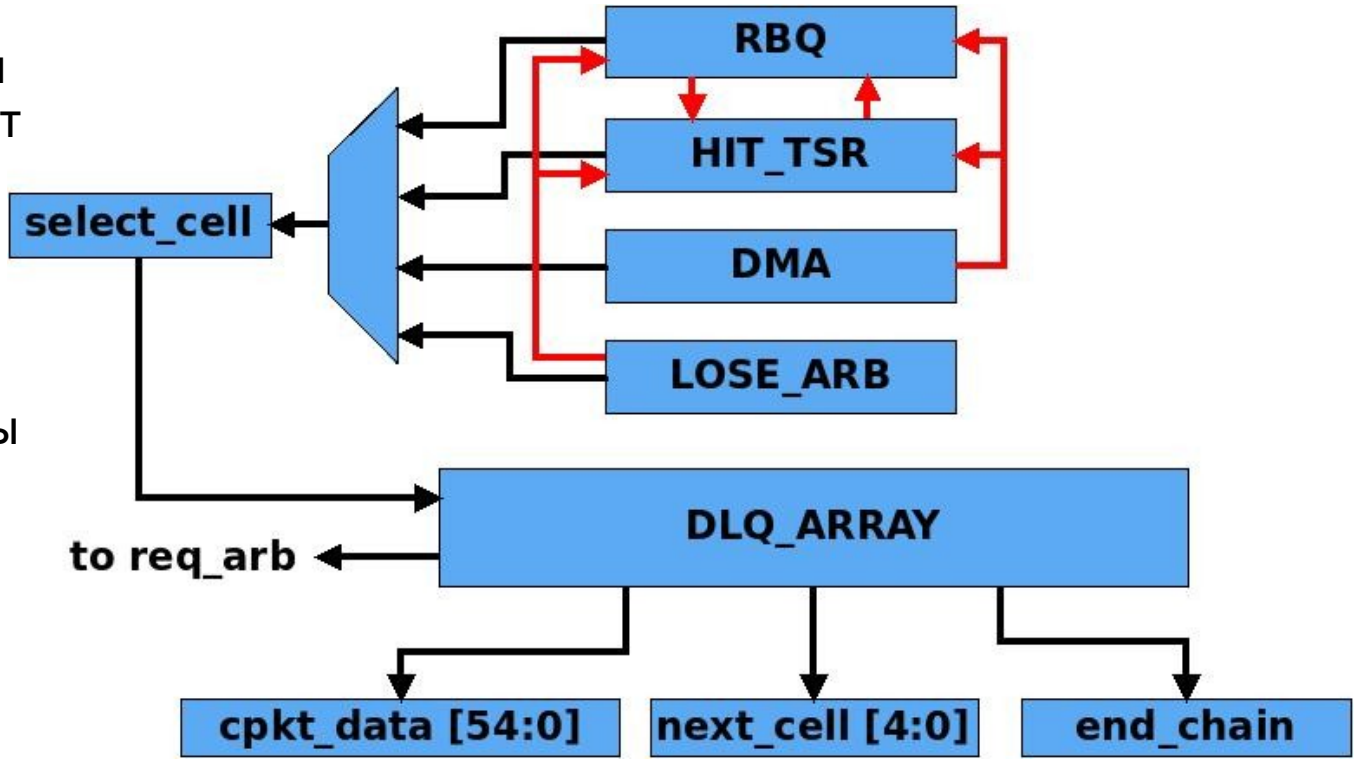


Буфер отложенных запросов: переходы между очередями

- При повторном прохождении конвейера, транзакция может снова получить блокировку. Для разрешения этих событий введены переходы между очередями DLQ.

→ Допустимые переходы

- Кроме обозначенных на рисунке переходов, возможны ещё следующие:
  - RBQ → RBQ
  - DMA → DMA



- При переходе транзакции из DMA в любую другую, все последующие DMA-транзакции отменяются и записываются в DMA-очередь. При этом очередь DMA-запросов блокируется до тех пор, пока не будет выполнен DMA-запрос, перешедший в другую очередь. Это гарантирует сохранения порядка выполнения DMA-транзакций.

Контроллер когерентности: физические параметры

Number of ports	1418
Number of nets	458287
Number of cells	454668
Number of combinational cells	394454
Number of sequential cells	60146
Number of macros/black boxes	68
Number of buf/inv	170726
Number of references	1893
Combinational area	319766
Buf/Inv area	105585
Noncombinational area	150485
Macro/Black Box area	1273055
Width, мкм	725
Length, мкм	3028
Total Area, мкм^2	2195300