YADRO ИМПУЛЬС

О группе разработки интерконнекта

В состав любого современного процессора помимо вычислительных ядер входит множество периферийных блоков. Например, к таким блокам относятся различные контроллеры внешних интерфейсов, контроллер прерываний и специализированные вычислители. Все эти устройства подключены к процессору с помощью системной шины. Блок, непосредственно реализующий функции системной шины, называется интерконнектом: он маршрутизирует запросы от вычислительных ядер к другим устройствам, и возвращает обратно их ответы. Производительность интерконнекта непосредственно влияет на производительность всей системы. Наша команда занимается разработкой и поддержкой интерконнектов для чипов, разрабатываемых командой YADRO.

Знания и навыки

Для того, чтобы продуктивно стажироваться в нашей группе, необходим следующий минимальный набор знаний и навыков:

- Базовые знания цифровой схемотехники
- Начальный опыт разработки RTL на Verilog/SystemVerilog и навыки работы с симулятором

Дополнительными плюсами будут:

- Опыт работы с ПЛИС
- Прочитал книгу "Цифровая схемотехника и архитектура компьютера, Дэвид Харрис и Сара Харрис"

Тестовое задание

Оценить ваши знания и навыки поможет тествое задание. Вы можете выбрать любой из перечисленных ниже вариантов.

NOTE	Сложность заданий отличается. Для собеседования достаточно решить один из вариантов. Мы оценим, если вы решите более сложный вариант.
NOTE	Разработанный блок может иметь задержку (latency), а может и не иметь - решать вам
NOTE	Мы оценим, если вы сделаете блок, обладающий минимальным latency и максимальной пропускной способностью. Но будет достаточно, если он просто будет работать. А вы отдельно (в README) перечислите ограничения вашей реализации и возможные способы их устранения.

Этапы выполнения задания

1. Проектирование. Результатом этапа является README, включающий:
    - описание принятых при проектировании решений;
    - (опционально) микроархитектурные диаграммы устройства;
    - описание работы спроектированных модулей;

      Нам важен в первую очередь ход ваших мыслей при проектировании блока.
2. Кодирование и bring-up тест. Результатом данного этапа является:
    - код на Verilog или System Verilog
    - testbench и набор тестовых векторов для среды ModelSim
    - (опционально) скрипт для запуска симулятора (Makefile или bash)

Вариант 1. Преобразователь valid/credit интерфейса в valid/ready интерфейс

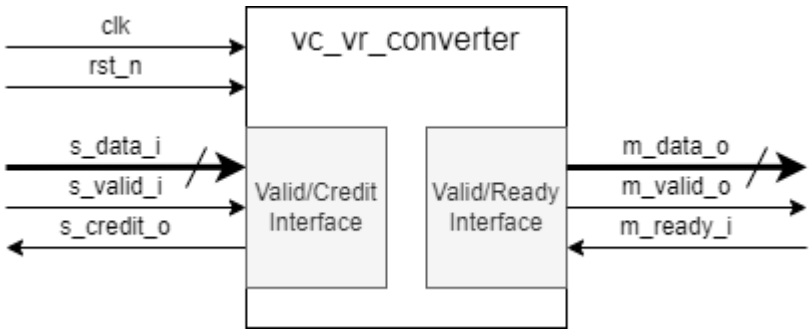

Figure 1. Структурная схема модуля vc_vr_converter

Интерфейс модуля

```
module vc_vr_converter #(
    parameter DATA_WIDTH = 8,
              CREDIT_NUM = 2
)(
    input  logic                    clk,
    input  logic                    rst_n,

    //valid/credit interface
    input  logic [DATA_WIDTH-1:0]   s_data_i,
    input  logic                    s_valid_i,
    output logic                    s_credit_o,

    //valid/ready interface
    output logic [DATA_WIDTH-1:0]   m_data_o,
    output logic                    m_valid_o,
    input  logic                    m_ready_i
);
```

Описание

Необходимо разработать преобразователь valid/credit интерфейса в valid/ready интерфейс.

Протокол приёма данных

1. На вход модуля поступают данные по шине `s_data_i` шириной `DATA_WIDTH`. Данные `s_data_i` передаются только в том случае, если `s_valid_i` равен единице.
2. Модуль имеет количество доступных кредитов равное `CREDIT_NUM`. Под кредитами здесь подразумевается количество данных `s_data_i`, которое способен принять модуль по сигналу `s_valid_i`. В том случае, если текущее количество кредитов в модуле равно нулю, то полученные в текущем такте данные `s_data_i` по сигналу `s_valid_i` будут потеряны. Подразумевается, что после снятия сброса с модуля текущее количество доступных кредитов равно параметру `CREDIT_NUM`.
3. Через порт `s_credit_o` передаётся информация о доступных кредитах. Импульс на 1 такт

на `s_credit_o` показывает master-устройству, что стал доступен ещё 1 кредит. При этом master-устройство не должно устанавливать сигнал `s_valid_i` в единицу при отсутствии доступных кредитов. Изначально, при активном сбросе модуля `vc_vr_converter`, master-устройство считает, что доступные кредиты отсутствуют. При снятии сброса модуль `vc_vr_converter` должен вернуть master-устройству CREDIT_NUM кредитов через порт `s_credit_o`.

Протокол передачи данных

1. На выход порта `m_data_o` передаются полученные данные через входной порт `s_data_i`. При этом данные на `m_data_o` считаются корректными только в том случае, если `m_valid_o` равен единице.
2. Данные `m_data_o` считаются переданными, если `m_valid_o` и `m_ready_i` одновременно установлены в единицу на текущем такте.
3. Если сигнал `m_valid_o` установлен в 1, то `m_data_o` не может менять свое значение, пока не будет подтверждения через `m_ready_i`.
4. Данные на выход должны передаваться в том же порядке, в каком они поступили на вход.

Ниже приведен пример преобразования интерфейсов для конфигурации:

```
DATA_WIDTH = 8
CREDIT_NUM = 2
```

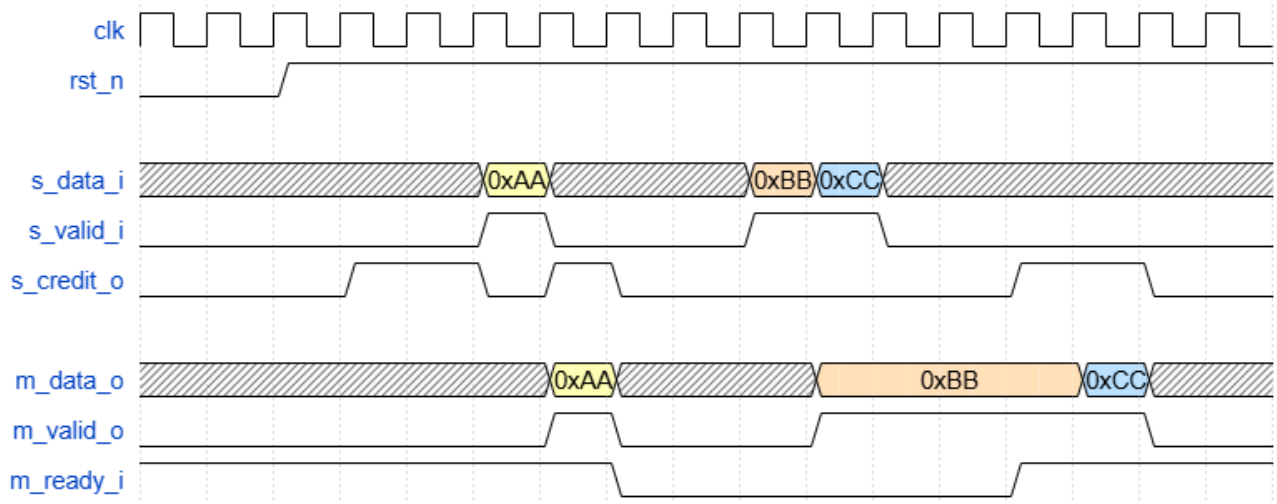

Figure 2. Пример преобразования интерфейсов

Вариант 2. Потоковый реордер-буфер

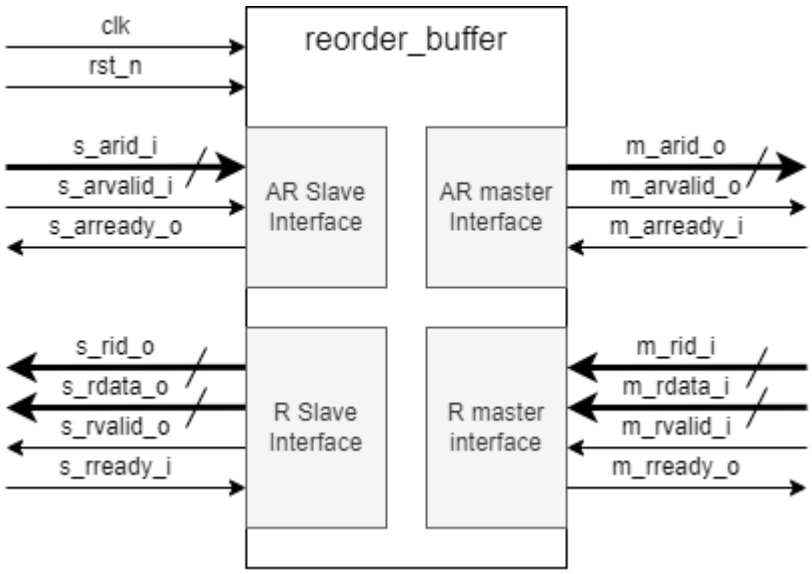

Figure 3. Структурная схема модуля reorder_buffer

Интерфейс модуля

```
module reorder_buffer #(
    parameter DATA_WIDTH  = 8
)(
    input  logic                    clk,
    input  logic                    rst_n,

    //AR slave interface
    input  logic            [3:0]   s_arid_i,
    input  logic                    s_arvalid_i,
    output logic                    s_arready_o,

    //R slave interface
    output logic [DATA_WIDTH-1:0]   s_rdata_o,
    output logic            [3:0]   s_rid_o,
    output logic                    s_rvalid_o,
    input  logic                    s_rready_i,

    //AR master interface
    output logic            [3:0]   m_arid_o,
    output logic                    m_arvalid_o,
    input  logic                    m_arready_i,

    //R master interface
    input  logic [DATA_WIDTH-1:0]   m_rdata_i,
    input  logic            [3:0]   m_rid_i,
    input  logic                    m_rvalid_i,
    output logic                    m_rready_o
);
```

Описание

Необходимо разработать потоковый реордер-буфер, который:

- Принимает поток из 16 уникальных ID в диапазоне от 0 до 15 через `AR slave` интерфейс. Порядок ID в потоке может быть произвольным.
- Передаёт поток ID без изменений в том же порядке из `AR slave` интерфейса в `AR master` интерфейс.
- Для каждого переданного ID на `AR master` интерфейсе принимает данные через `R master` интерфейс. Каждое слово данных также помечено ID, но порядок приёма данных на `R master` интерфейсе может не совпадать с порядком передачи ID на `AR master` интерфейсе.
- Выдает данные на `R slave` интерфейс, переупорядочивая их таким образом, чтобы ID потока данных соответствовали порядку ID из потока ID на `AR slave` интерфейсе.

Протокол приёма ID

1. На вход интерфейса `AR slave` на шину `s_arid_i` модуля reorder_buffer поступают ID в диапазоне от 0 до 15. При этом ID на шине `s_arid_i` считаются корректными только в том случае, если `s_arvalid_i` равен единице.
2. ID на шине `s_arid_i` считается принятым, если `s_arvalid_i` и `s_arready_o` одновременно установлены в единицу на текущем такте.
3. Если сигнал `s_arvalid_i` установлен в 1, то `s_arid_i` не может менять свое значение, пока не будет подтверждения через `s_arready_o`.

Протокол передачи ID

1. На выход интерфейса `AR master` на шину `m_arid_o` модуля reorder_buffer без изменений и в том же порядке передаются ID, полученные по интерфейсу `AR slave`. При этом ID на шине `m_arid_o` считаются корректными только в том случае, если `m_arvalid_o` равен единице.
2. ID на шине `m_arid_o` считается принятым, если `m_arvalid_o` и `m_arready_i` одновременно установлены в единицу на текущем такте.
3. Если сигнал `m_arvalid_o` установлен в 1, то `m_arid_o` не может менять свое значение, пока не будет подтверждения через `m_arready_i`.

Протокол приёма данных

1. Для каждого переданного ID через интерфейс `AR master` на вход интерфейса `R master` на шину `m_rid_i` возвращается тот же ID вместе с данными на шине `m_rdata_i` шириной DATA_WIDTH. При этом данные вместе с ID могут приходить с произвольной задержкой и порядок ID на шине `m_rid_i` в общем случае может не соответствовать порядку передачи ID по интерфейсу `AR master`. Данные на `m_rdata_i` и ID на `m_rid_i` считаются корректными только в том случае, если `m_rvalid_i` равен единице.
2. Данные `m_rdata_i` и ID `m_rid_i` считаются принятыми, если `m_rvalid_i` и `m_rready_o` одновременно установлены в единицу на текущем такте.
3. Если сигнал `m_rvalid_i` установлен в 1, то `m_rdata_i` и `m_rid_i` не могут менять свое значение, пока не будет подтверждения через `m_rready_o`.

Протокол передачи данных

1. Если на интерфейсе `R master` были получены данные с ID, который был принят раньше всех на интерфейсе `AR slave`, то эти данные передаются на выход порта `s_rdata_o`, а на выход `s_rid_o` передаётся ID, который соответствует этим данным. Остальные данные вместе с ID должны передаваться аналогично в порядке следования ID на интерфейсе `AR slave`. При этом данные на `s_rdata_o` и ID на `s_rid_o` считаются корректными только в том случае, если `s_rvalid_o` равен единице.
2. Данные `s_rdata_o` и ID `s_rid_o` считаются переданными, если `s_rvalid_o` и `s_rready_i` одновременно установлены в единицу на текущем такте.
3. Если сигнал `s_rvalid_o` установлен в 1, то `s_rdata_o` и `s_rid_o` не могут менять свое значение, пока не будет подтверждения через `s_rready_i`.

Ниже приведен пример реордеринга для четырех ID от 0 до 3 для конфигурации:

```
DATA_WIDTH = 8
```

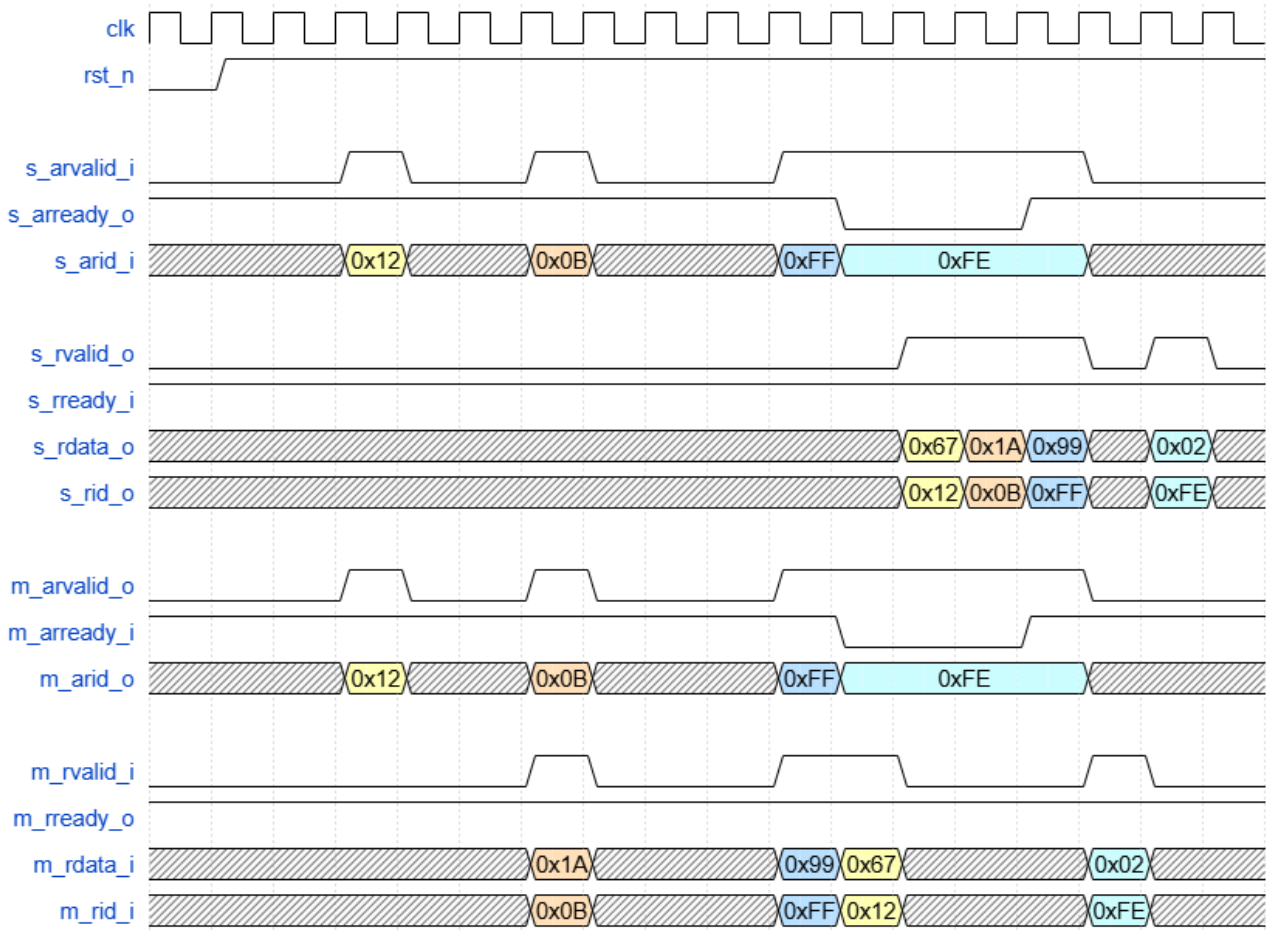

Figure 4. Пример реордеринга