Практическое Задание №5

Практическое задание №5 состоит из 10 упражнений: 6 обязательных и 4 опциональных на дополнительную оценку.

Упражнения посвящены трём темам — FIFO, конвейеризованным блокам вычислений, и процессорному ядру schoolRISCV.

Структура папок в текущем практическом задании следующая:

- `05_01_fifo_with_counter_baseline` - базовый пример FIFO для ознакомления
- `05_02_fifo_pow2_depth` - первое упражнение с FIFO
- `05_03_fifo_empty_full_optimized` - второе упражнение с FIFO
- `05_04_fifo_with_reg_empty_full` - третье упражнение с FIFO
- `05_05_a_plus_b_using_fifos_and_double_buffer` - упражнение с формулой `a + b`
- `05_06_sqrt_formula_pipe` - упражнение с формулой 2 из ПЗ №4 и модулем `isqrt`
- `05_07_cpu_baseline` - базовый пример schoolRISCV для ознакомления
- `05_08_cpu_with_comb_mul_instr` - упражнение на добавление инструкции умножения
- `05_09_cpu_mul_with_latency` - опциональное упражнение
- `05_10_cpu_with_b_instr` - опциональное упражнение
- `05_11_cpu_fetch_with_latency` - опциональное упражнение
- `05_12_three_cpus_sharing_instr_memory` - опциональное упражнение

В каждом упражнении есть комментарий `// Example` с кодом для примера,
и комментарий `// Task`, рядом с которым нужно расположить код вашего решения.

Так же, в каждом упражнении есть файл `tb.sv`, который осуществляет минимальную проверку работоспособности вашего решения.

Предисловие

В процессе работы над решениями, возможно запускать проверку каждого отдельного упражнения с помощью соответствующего скрипта `run_using_iverilog` в каждой из директорий. Так же можно запустить скрипт в корневой директории ПЗ 5 для проверки всех упражнений.

В файле `tb.sv` любого из упражнений можно раскомментировать строку `$dumpvars;` для генерации `dump.vcd` файла. Можно воспользоваться командой `gtkwave dump.vcd` для просмотра файла, либо раскомментировать соответствующую строку в файле скрипта `run_all_using_iverilog`.

Упражнение 1. FIFO с глубиной кратной степени двойки

> Упражнение: Обязательное
>
> Директория: `05_02_fifo_pow2_depth`

Задание: Реализовать недостающий код обновления расширенного указателя чтения и сигнал `empty` для полноценного функционирования FIFO.

Упражнение 2. Оптимизированное FIFO

> Упражнение: Обязательное
>
> Директория: `05_03_fifo_empty_full_optimized`

Оптимизация FIFO заключается в отсутствии счётчика зависящего от глубины, но наличие только 2 битов для определения взаимного расположения указателей чтения и записи.

Задание: Реализовать код обновления указателя чтения и чётности круга у соответствующего бита, а так же сформировать сигнал `full` с учётом констант `equal_ptrs` и\или `same_circle`.

Упражнение 3. FIFO с регистрами empty и full

> Упражнение: Обязательное
>
> Директория: `05_04_fifo_with_reg_empty_full`

В данном упражнении сигналы empty и full являются регистрами, внутренние сигналы же формируются комбинационно и далее записываются в соответствующие регистры.

Задание: Реализовать код формирования комбинационного сигнала `rd_ptr_d`, а так же описать логику формирования сигналов `empty_d` и `full_d` при выставлении сигнала `pop`

Упражнение 4. Формула a + b с FIFO

> Упражнение: Обязательное
>
> Директория: `05_05_a_plus_b_using_fifos_and_double_buffer`

В данном упражнение реализуется схема примерно описанная в статье "FIFO для самых маленьких" в Примере 3. Две FIFO на входах операции сложения выравнивают результаты по времени и результат складывается в двойной буфер на выходе формулы.

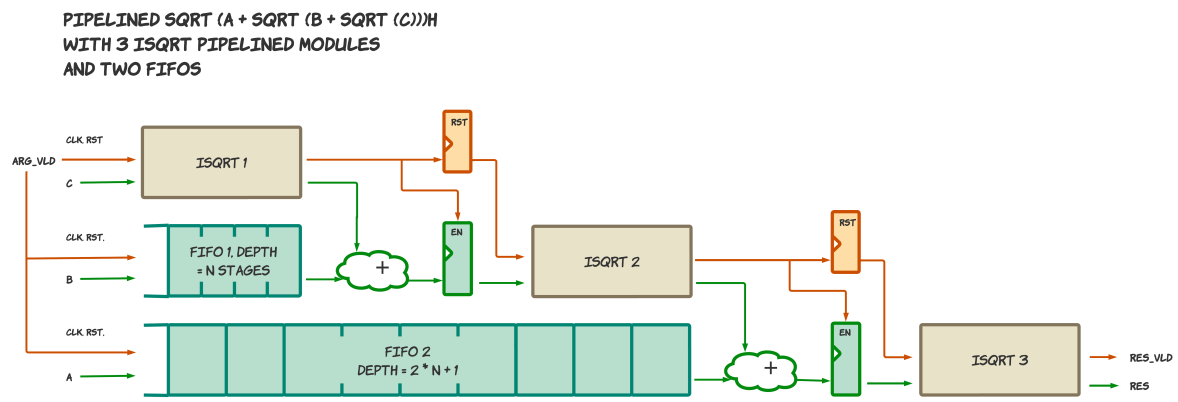
Задание: Соединить все внутренние модули упражнения используя, в том числе, внешние сигналы valid/ready. Заготовки всех связываний (assign) приведены в комментариях `Task`, необходимо реализовать логику формирования сигналов после `=`

Упражнение 5. Formula 2 с FIFO

> Упражнение: Обязательное
>
> Директория: `05_06_sqrt_formula_pipe`

Структура папки упражнения идентична упражнению с формулами в Практическом Задании 4. Рекомендуется ознакомиться со статьей Юрия Панчула "Что умеют и не умеют писать на SystemVerilog для ASIC и FPGA американские студенты?" в журнале FPGA-Systems Magazine.

Задание: Реализовать последний описанный в статье случай — вычисление Формулы 2 используя конвейеризованные модули `isqrt` и готовый модуль из файла `flip_flop_fifo_with_counter`.



Упражнение 6. Новая инструкция умножения в процессоре

> Упражнение: Обязательное
>
> Директория: `05_08_cpu_with_comb_mul_instr`

В данном упражнении, в первую очередь необходимо ознакомиться со структурой schoolRISCV процессора в папке `05_07_cpu_baseline`. В текущем и последующих упражнениях базовая структура процессора похожая, но расширяется в зависимости от условий упражнения. Так же желательно ознакомиться с лабораторной работой `30_schoolriscv` в репозитории `basics-graphics-music` и посмотреть 21 занятие Школы Синтеза Цифровых Схем, где подробно объясняется как работать с проектом schoolRISCV.

Так же, полезно ознакомиться со стандартом RISC-V (RV32I User-Level ISA и расширением RV32M) и расширять процессор в соответствии со стандартом.

Задание: Необходимо добавить инструкцию умножения `mul` в процессор для корректной работы `program.s` программы по вычислению чисел Фибоначчи. Для этого необходимо обновить файл `sr_cpu.svh` добавлением корректных констант, а так же файл АЛУ `sr_alu.sv`.

Упражнение 7. Инструкция умножения с задержкой

> Упражнение: **Дополнительное**
>
> Директория: `05_09_cpu_mul_with_latency`

Задание: Расширить предыдущее упражнение и модифицировать процессор для корректной работы с задержкой результата операции умножения на 1 такт.

Упражнение 8. Новая инструкция безусловного перехода `b`

> Упражнение: **Дополнительное**
>
> Директория: `05_10_cpu_with_b_instr`

Задание: Добавить инструкцию безусловного перехода `b`. Для этого необходимо расширить интерфейс модулей `sr_control.sv` и `sr_decode.sv`, а так же добавить корректную логику обновления программного счётчика `pcNext` и записи результата инструкции в регистр `wd3`.

Упражнение 9. Модуль памяти инструкций с задержкой

> Упражнение: **Дополнительное**
>
> Директория: `05_12_three_cpus_sharing_instr_memory`

Задание: Модифицировать процессор для корректной работы с задержкой получения инструкции из памяти инструкций в 1 такт.

Возможные решения:

- Добавить дополнительный флаг `imDataVld` обозначающий валидность данных, приходящих из "instruction memory"
- Реализовать конвейерную работу всего процессора (решение повышенной сложности)

Упражнение 10. Три процессора и арбитр

> Упражнение: **Дополнительное**
>
> Директория: `05_12_three_cpus_sharing_instr_memory`

Ознакомиться с работой арбитра в файле `round_robin_arbiter_8.sv`

Задание: Реализовать кластер процессоров в файле `cpu_cluster.sv` состоящий из 3-х экземпляров ядра schoolRISCV, разделяющих одну память инструкций.