Лабораторная работа №3

Построение схем

Инструментарий и требования к работе

ПО	Logisim-evolution 3.8.0 и Icarus Verilog 12
Примечания по сложностям	Каждая часть работы может быть выполнена на любой из сложностей. Чтобы работа была принята на проверку, необходимо сделать как минимум `lite` реализацию на Logisim и одну из `lite` реализаций на SystemVerilog.

Задание

Работа состоит из трёх частей: моделирования в Logisim и описания схемы в двух вариантах на Verilog. В заданиях задана одна и та же схема.

Схема

Собрать схему "Синхронный стек". Одна ячейка хранит 4 бита. У стека нет состояния "занятости" (занято/свободно). Синхронизация `CLK` должна работать по высокому уровню (аналогично рассмотренному на занятии JK-триггеру).

Схема должна обрабатывать следующие команды (подаются через вход команд `COMMAND`):

Код	Команда	Пояснение
`0`	`nop`	Нет операции, ничего не происходит.
`1`	`push`	Вершина стека смещается на следующую ячейку, затем в текущую ячейку записывается указанное значение.
`2`	`pop`	Снять значение ячейки с вершины стека: подать его на выход и сместить вершину стека. Значения в ячейках не меняются.
`3`	`get`	Получить значение ячейки по индексу относительно вершины стека.

Вершина стека (и сумма его с индексом в команде `get`) всегда берётся по модулю размера стека.

Отдельно должен быть вход для инициализации (`RESET`) – асинхронно переводит схему в начальное состояние и обнуляет состояние внутренней памяти.

Варианты

`lite`	`normal`
Размер стека (число ячеек) – 8	Размер стека (число ячеек) – 5
Должны быть отдельно вход `I_DATA` и выход `O_DATA` данных	Должен быть вход-выход `IO_DATA` данных

При выполнении normal-версии обратите внимание, чтобы у вас не возникало короткого замыкания на входо-выходах (когда на них подаётся значение и снаружи, и из вашей схемы).

Задание 1. Logisim

Собрать описанную схему на Logisim. В репозитории выданы проекты с прототипом схемы. Модифицировать схемы *main* и уже размещённые контакты в проекте запрещено.

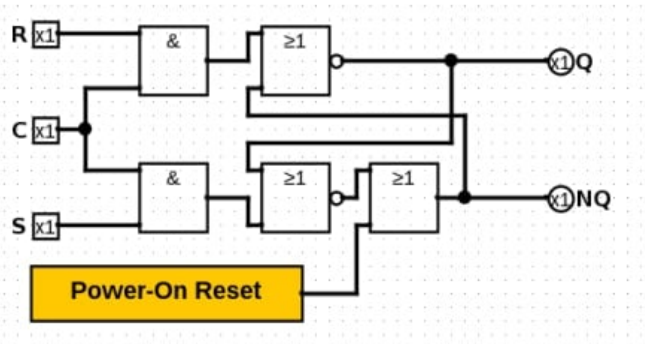
Требуемая схема должна быть реализована на подсхеме *stack*, подсхема *main* должна содержать только один экземпляр подсхемы *stack* и элементы ввода и вывода, для тестирования собранной схемы. Необходимо использовать дополнительные подсхемы (например, подсхема ячейки памяти) для реализации подсхемы *stack*. Файл с проектом (всеми подсхемами) должен называться *stack_logical_(lite/normal).circ*. Если реализуется базовый вариант (отдельно вход и отдельно выход данных, то нужно дополнять проект с суффиксом "*stack_logical_lite.circ*", иначе – "*stack_logical_normal.circ*").

В задании можно использовать только передаточный вентиль, полевые транзисторы и базовые логические элементы. Из вспомогательных элементов: разветвители (splitter), датчики (probe), тоннели (tunnel). Соответственно все триггеры, мультиплексоры и пр. собираются самостоятельно в виде подсхем. Также стоит собрать часто повторяющиеся элементы в отдельные подсхемы.

Направление всех логических элементов: Восток. Исключение – элементы выхода, для них направление Запад. Входо-выходы располагаются снизу. Все входы и выходы подсхем – типа pin (контакт).

Все входы, выходы и подсхемы должны быть названы (иметь заполненный label) и адекватно называться.

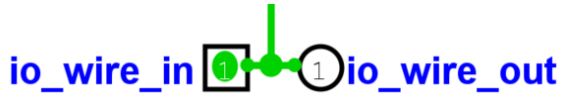
Необходимо использовать элемент POR (в русской локализации сигнал сброса). Нужен для багфикса симуляции Logisim при сборке триггеров (если проблема ещё возникает). Обратите внимание, что в Verilog POR не нужен и не используется.



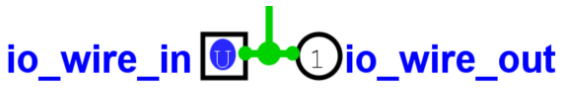
Пояснение про входо-выход

Входо-выходы на схеме можно подключить к pin (контакту) и probe (датчику). Это позволит задавать значение через pin, когда провод используется как вход, и наблюдать значение, когда провод используется как выход.

В случае записи контакт, подсоединённый с inout проводу, ставится в 0 или 1.

В случае чтения данных с провода необходимо установить на контакте высокоимпедансное состояние (значение U).

Задание 2. SystemVerilog

Собрать описанную схему на SystemVerilog**.**

Задание 2 делится на 2 подзадания. Оба подзадания реализуют описанную ранее схему двумя разными способами. К обоим заданиям написан тестирующий модуль (testbench), при запуске которого будет проводиться тестирование ваших описанных модулей.

В интерфейсе всех ваших модулей сначала объявляются выходы, затем входо-выходы, затем входы.

Если реализуется базовый вариант (отдельно вход и отдельно выход данных, то модули называются с суффиксом "_lite", иначе – "_normal"). В репозитории выданы прототипы модулей и тестовых модулей. Модифицировать модули (название и интерфейс) в шаблонах запрещено.

Задание 2.1. Описание через структурную модель

Для реализации стека нужно использовать только встроенные примитивы: транзисторы (pmos, nmos, cmos) и базовые логические элементы. Операторы (`!`, `|`, `&`, `~` , `-` и пр.) использовать в структурной модели нельзя. Типы данных: `wire`.

Модуль, реализующий стек, должен называться *stack_structural_[lite/normal]*, тестирующий модуль – *stack_structural_tb*.

Файл, содержащий все модули этого задания должен называться *stack_structural.sv*, файл с *stack_structural_tb* – *stack_structural_tb.sv*. Шаблоны приведены в репозитории.

Задание 2.2. Описание через поведенческую модель

Для реализации стека нужно использовать операторы поведенческого моделирования (`always`, `initial`, `assign`, операторы, …) и управляющие конструкции (`case`, `if`, …). Типы данных: `reg`, `wire`.

Модуль, реализующий стек, должен называться *stack_behaviour_[lite/normal]*, тестирующий модуль – *stack_behaviour_tb*. Файл, содержащий все модули этого задания должен называться *stack_behaviour.sv*, файл с *stack_behaviour_tb* – *stack_behaviour_tb.sv*. Шаблоны приведены в репозитории.

Полезные материалы по SystemVerilog:

раздел Массивы в “Справочная информация о Verilog” *или* *тык* *или* *тык*