федеральное государственное автономное образовательное учреждение
высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ОТЧЕТ

по лабораторной работе №2

по дисциплине «**Функциональная схемотехника**»

Вариант 9

Автор: Кулаков Н. В.

Факультет: ПИиКТ

Группа: Р33312

Преподаватель: Васильев С.Е.

Санкт-Петербург 2023

Оглавление

1. Задание

1.1. Описание

Лабораторная работа посвящена проектированию последовательностной логики на уровне регистровых передач с использованием языка описания аппаратуры Verilog HDL. В первой части работы предлагается разработать несколько простых блоков цифровой последовательностной логики и объединить их для выполнения заданной функции в одно функционирующее устройство.
Во второй части работы предлагается разработать устройство, управляющее входным потоком данных с помощью одного из указанных алгоритмов обработки.

1.2. Таблица варианта

Вариант	Функция 1	FSM	Функция 2	Разрядности	Делит. Частоты
9	COUNT_FREE	FSM_4	LRU	32 бита	5

2. Выполнение

2.1. Счетчик

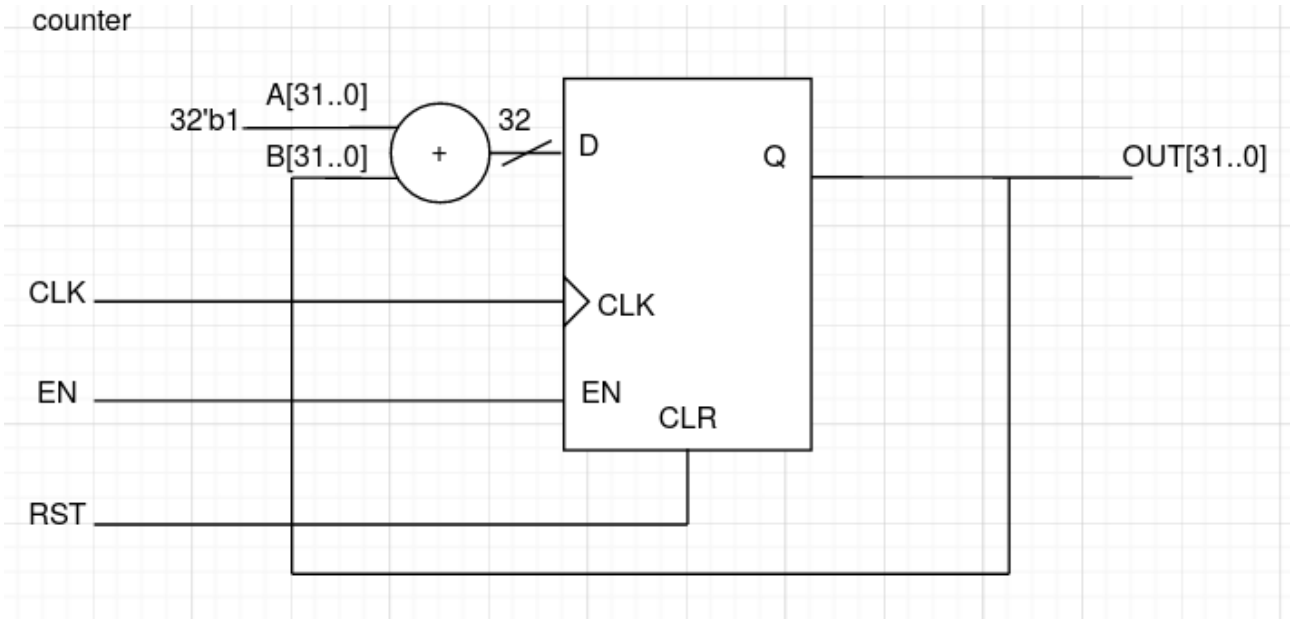

Рисунок 1: Синхронный счетчик по переднему фронту с асинхронным сбросом 32 разряда.

Счетчик тактируется от CLK по переднему фронту. Когда уровень EN высокий, то счетчик инкрементируется, когда уровень низкий — счетчик сохраняет свое значение с предыдущего такта. По переднему фронту RST счетчик может сбросить свое значение на значение на 0.

На выходе по переднему фронту выставляется значение Q(t-1)+1 по модулю разрядности счетчика, так как он построен на динамическом D триггере по переднему фронту.

2.2. Сдвиговый регистр

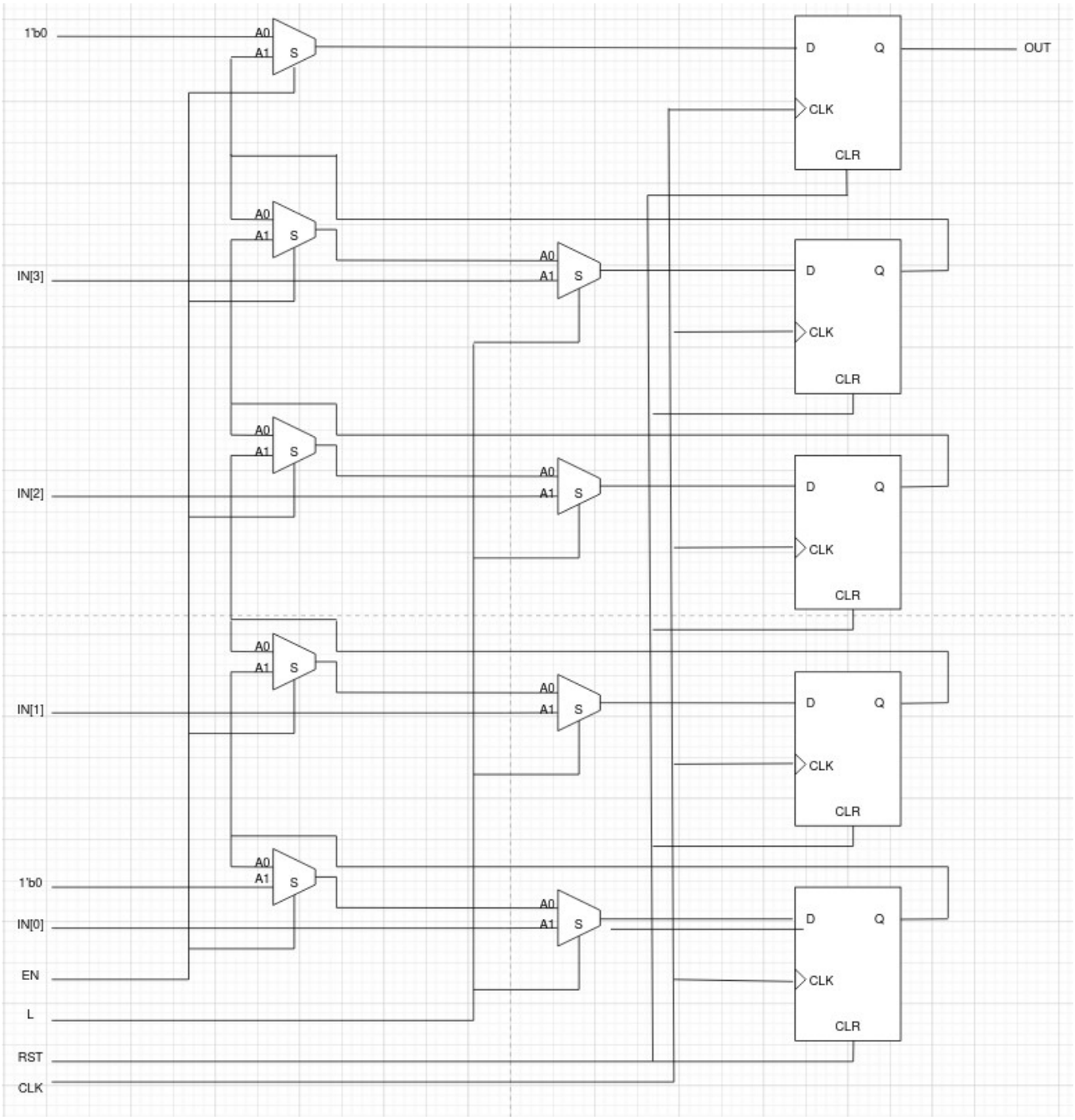

Рисунок 2: Синхронный сдвиговый регистр с входом разрешения работы, загрузки и асинхронным сбросом на 4 разряда

Сдвиговый регистр тактируется по переднему фронту. Асинхронно при установлении флага RST сдвиговый регистр и OUT сбрасываются. При установлении флага EN начинает схема начинает сдвигать значения влево (к старшим разрядам), а то, что не помещается в старший разряд, записывается в

регистр для бита переноса. При установлении L осуществляется загрузка в сдвиговый регистр. Заметим, что OUT в этом случае не обнулся.

2.3. Конечный автомат (FSM_4)

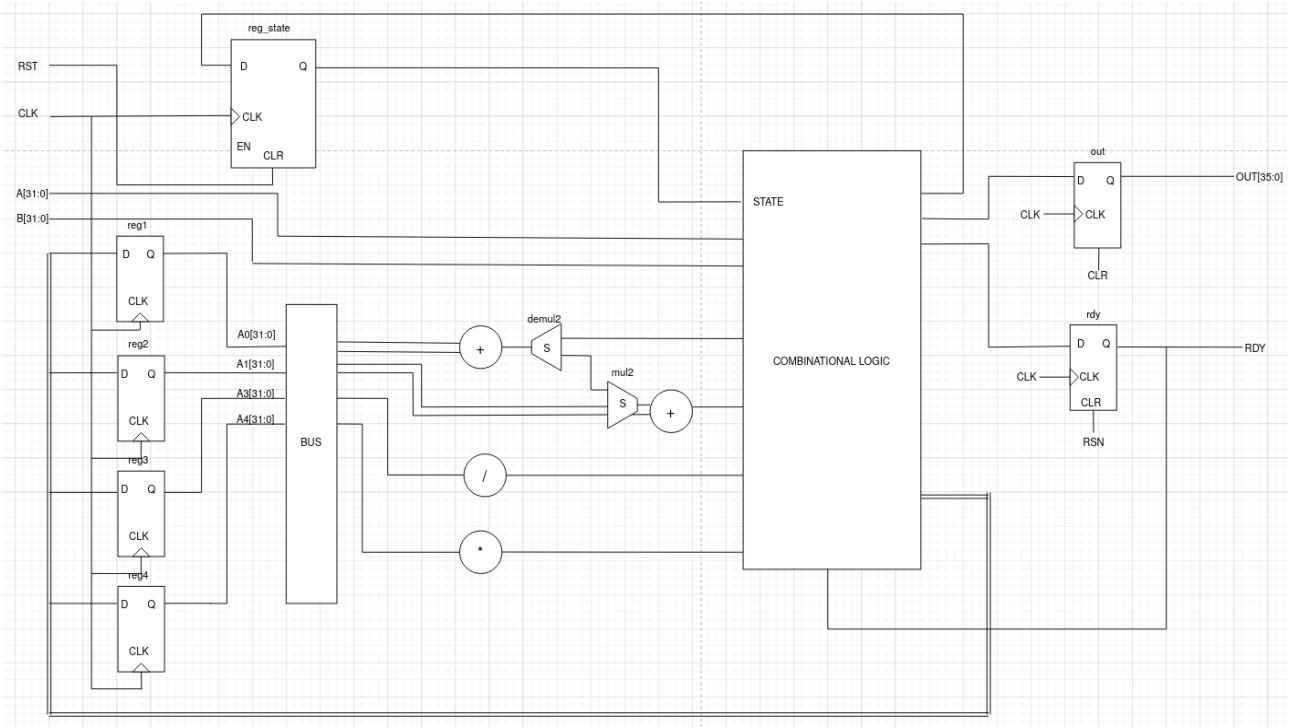

Рисунок 3: Синхронный конечный автомат с асинхроный сбросом в начальное состояние

Для установки конечный автомат в начальное состояние необходимо подать сигнал RST на вход. Затем, переходя из состояние в состояние, высчитывается требуемая функция. Когда конечный автомат закончит свою работу, он перейдет в начальное состояние и останется в нем до тех пор, пока он не будет сброшен подачей RST. Когда автомат посчитает свое значение, на выходе OUT он выставит результат и установит RDY в 1.

2.4. Делитель частоты

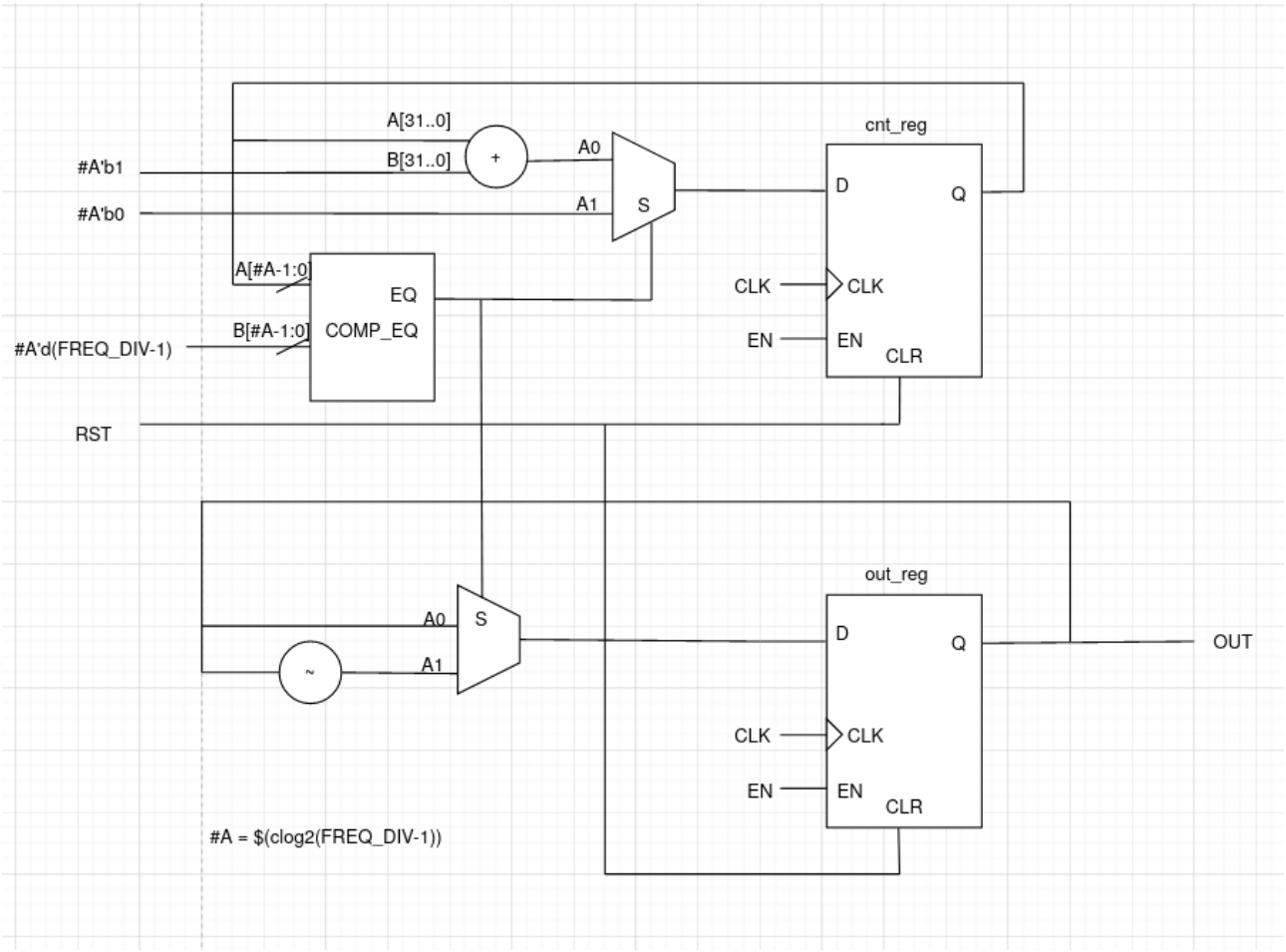

Рисунок 4: Синхронный делитель частоты на FREQ_DIV с асинхронным сбросом

При подаче асинхронного сигнала сброса FST делитель частоты сбрасывается. С каждым тактом инкрементируется значение регистра cnt_reg до тех пор, пока оно не станет равным значению FREQ_DIV-1. Тогда инвертируется значение регистра out_reg и обнуляется значение cnt_reg. Выход делителя частоты — OUT.

2.5. Первая функция (COUNT_FREE)

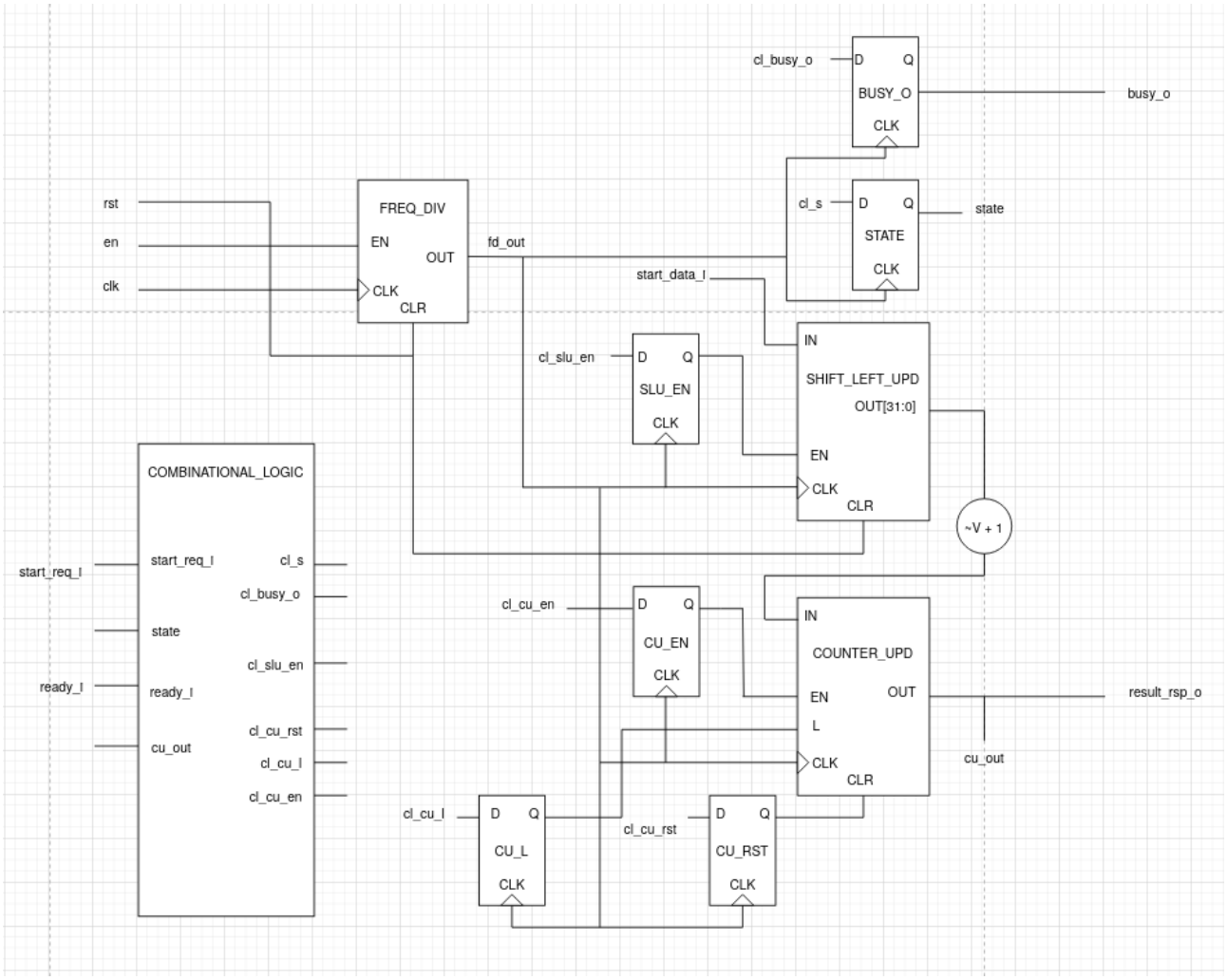

Рисунок 5: Синхронная функция COUNT_FREE с асинхронным сбросом, тактирующаяся от FREQ_DIV

Данная функция реализована как конечный автомат мура. Для реализации используются регистры CU_L, CU_RST, CU_EN для счетчика, с параллельным входом и выходом флага переноса; SLU_EN для сдвигового регистра с последовательным входом и параллельным выходом; STATE — регистр состояния, BUSY_O — для флага занятости. FREQ_DIV — делитель частоты, от которого тактируются все компоненты COUNT_FREE.

Пример работы: читаем пока на входе есть данные, с каждым тактом на данном этапе в счетчик записывается текущий результат на сдвиговом регистре (потому что не получится сделать после чтения, так как реализован как конечный автомат мура, получается будет лишний такт, что не даст реализовать нулевое

ожидание), к которому применена операция нахождения доп-кода. Счетчик начинает инкрементироваться, и выставится бит o_result_rsp когда сформируется бит переноса. Комбинационная логика спрятана за компонентом COMBINATIONAL_LOGIC.

2.6. Сложно-функциональный блок (BUFFER_LRU)

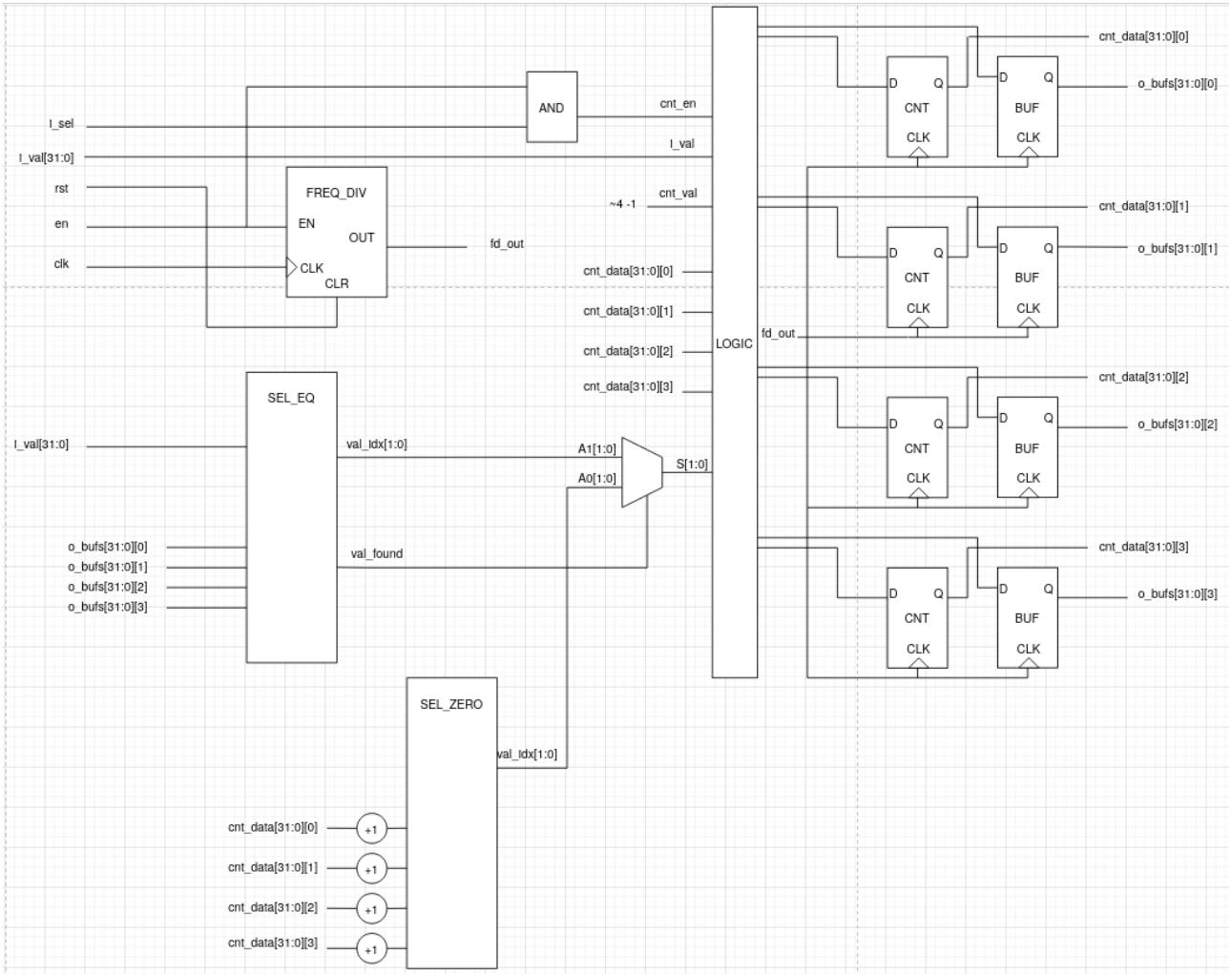

Рисунок 6: Синхронный BUFFER_LRU с асинхронным сбросом на 4 буфера

По заданию требовалось реализовать синхронный BUFFER_LRU с асинхронным сбросом на 8 буфером. На рисунке представлен для 4 со скрытой комбинационной логикой выбора значения на запись в буфер для поддержания состояния системы. На вход модулю поступают сигналы i_sel — на входе есть запрос на запись в буфер, i_val — само значение на запись, en — вход «работы», rst — вход асинхронного сброса, clk — такты. На выходе массив o_bufs, в

котором содержатся все значения буферов без учета присутствия (для таких в буфере записано значение по умолчанию). На рисунке не показан асинхронный сброс регистров счетчика (по которым выбирается буфер, в который требуется записывать данные) и установкой значений для счетчиков в силу громоздкости всей схемы.

Алгоритм таков: за каждым буфером присутствует счетчик, в который мы записываем тогда и только тогда, когда значение в нем плюс один равно нулю. Не может возникнуть ситуации, когда такой результат получится у нескольких счетчиков сразу, так как состояние поддерживается с установки значений по умолчанию.

Таким образом выбор буфера осуществляется по следующему алгоритму:

1) Проверяем, записано ли уже такое значение в буфер. Если да, то сбрасываем время счетчика, на такое, что в случае, когда значение не будет найдено, выбор данной ячейки будет осуществлен в последнюю очередь при дальнейших запросах (cnt_data[i] <= ~BUF_SIZE + 1). Инкрементируем те счетчики, значение которых меньше, чем то, которое было записано в cnt_data[sel_idx], sel_idx — выбранный номер буфера).

2) Если такой ячейки не существуем, то выбираем буфер, к которому реже всего было выполнено обращение, а это тот, у которого значение в счетчике плюс один равно 0. Сбрасываем значение привязанного счетчика к буферу в ~BUF_SIZE + 1, и всех остальных, то есть меньших, поскольку наше счетчик имеет наибольшее значение, инкрементируем на единицу.

Если не установлен en и i_sel, то инкрементирование счетчиков не происходит, и буферы не изменяют своих значений.

3. Выводы

Я устал и узнал много нового.