Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление системного и прикладного программного обеспечения

# ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

**курса «Функциональная схемотехника»**

**по теме: «Прототипирование цифровой схемы с помощью ПЛИС»**

**Вариант № 6**

Выполнил студент:

Тюрин Иван Николаевич

группа: P33102

Преподаватель:

Кустарев П. В.,

Васильев С.Е.

Санкт-Петербург, 2024 г.

# Содержание

# Лабораторная работа № 3 Прототипирование цифровой схемы с помощью ПЛИС

## 1. Введение

Лабораторная работа посвящена прототипированию разработанных модулей на FPGA. В качестве задания необходимо будет разместить сложнофункциональный блок, разработанный вами в второй лабораторной работе, на специализированной плате, подключив к нему часть доступных периферийных устройств

## 2. Задание варианта № 6

Для реализации был выбран блок из второй части лабораторной работы - Арбитр, Очередь FIFO/LIFO, Буфер MRU/LRU.

Для подачи запросов в ваш модуль следует использовать кнопки и переключатели на плате. Для вывода информации следует использовать семисегментные индикаторы и светодиоды. В каждом варианте должен быть реализован сигнал асинхронного сброса и его активация должна быть привязана к кнопке на плате.

### 2.1. Модуль «Arbiter»

Для передачи нескольких потоков транзакций необходимо организовать потактовую симуляцию с использованием кнопки. Одно нажатие на кнопку должно симулировать один тактовый период. Так как потоков всего четыре, а переключателей на плате 16, то каждому потоку отводится 3 переключателя на биты шины данных и один переключатель на бит валидности данных. При выдаче транзакции арбитром, на одной группе семисегментных индикаторов должен отобразиться номер потока, а на другой группе - переданные данные. На светодиодах необходимо отображать сигналы “ready” к

каждому потоку. Если в текущий такт данных не было, то семисегментные индикаторы должны быть погашены.

# 3. Выполнение задания

## 3.1. Архитектура верхнего модуля

В начале выполнения работы была разработана архитектура требуемого модуля, рис. 1.1, для этого пришлось изучить возможности платы Nexys A7 100T. Ключевыми особенностями были интерфейс взаимодействия с 7-сегментным дисплеем с помощью катодов и анодов и обработка сигнала кнопок. Главную действующую роль выполняет модуль арбитра, а драйвер дисплея лишь правильным образом распределяет сигналы арбитра во времени для подачи на порты дисплея. Состояние арбитра (сигнал валидности и занятости) подаются на пару RGB-светодиодов.

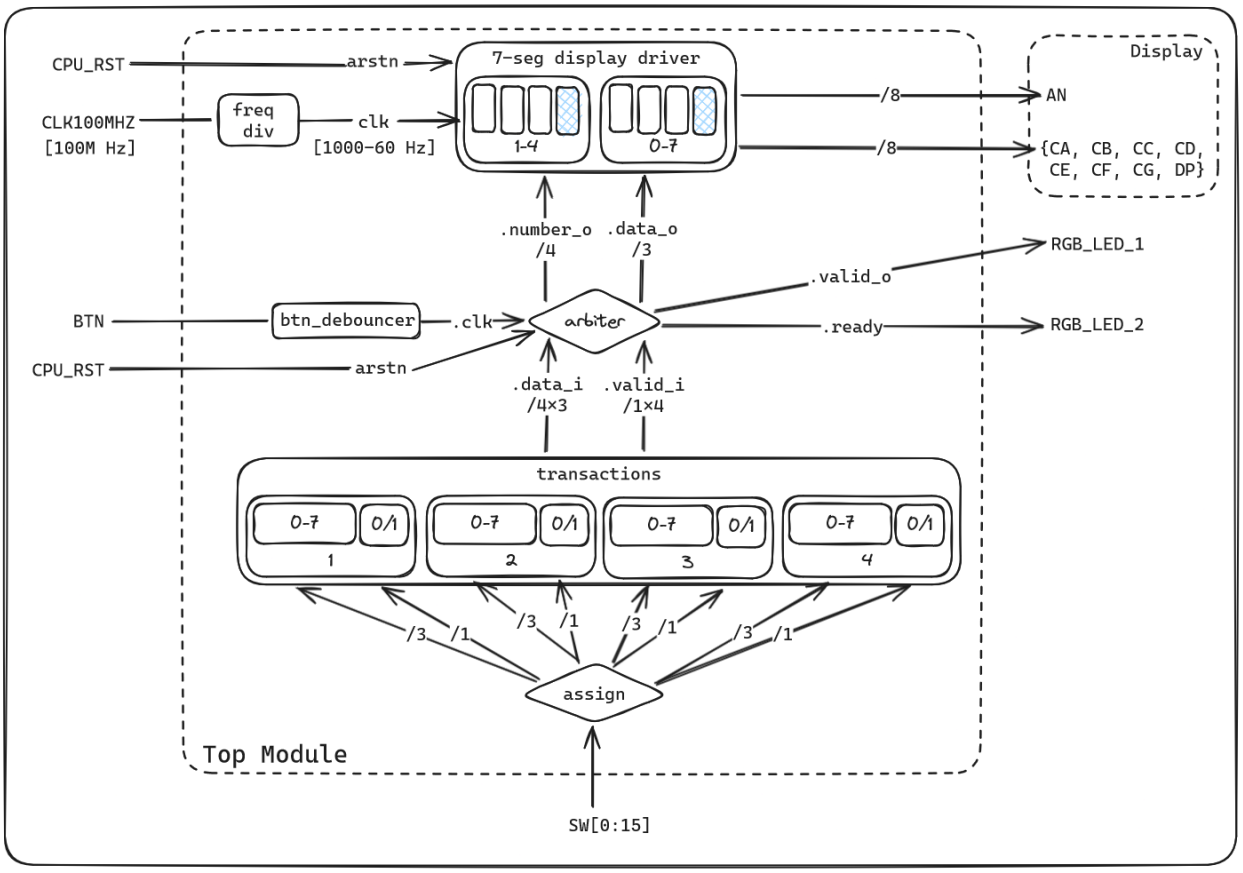


Рис. 1.1: Архитектура верхнего модуля

Особого внимания требует механизм обработки входного сигнала с различных механических элементов платы, таких как кнопки и сдвиговые переключатели. В силу их внутреннего устройства, в процессе нажатия и отпускания может происходить дребезг контактов, из-за чего возникает высокочастотный сигнал, который может спровоцировать ложное срабатывание. На сдвиговых переключателях дребезга контактов не было замечено и поэтому модуль защиты использовался только для нажимных кнопок. Более того, в ходе разработки этого модуля выло выяснено, что ему требуется

сигнал сброса, и поэтому он не может быть использован для подавления дребезга на кнопке сброса ЦПУ, да оно и не нужно — процессор в какой-то момент будет сброшен в последний раз.

## 3.2. Переключатели платы ПЛИС

Для разработки модуля защиты от дребезга пришлось изучить примеры решений представленных в различных источниках. Все они работают по-разному, но придерживаются одного принципа — перенос входного сигнала с одной частоты на другую, т.е. считывание промежуточных значений с некоторым промежутком времени и работа именно с ними. Графики показывающие описанный принцип работы можно видеть на рис. 1.2.

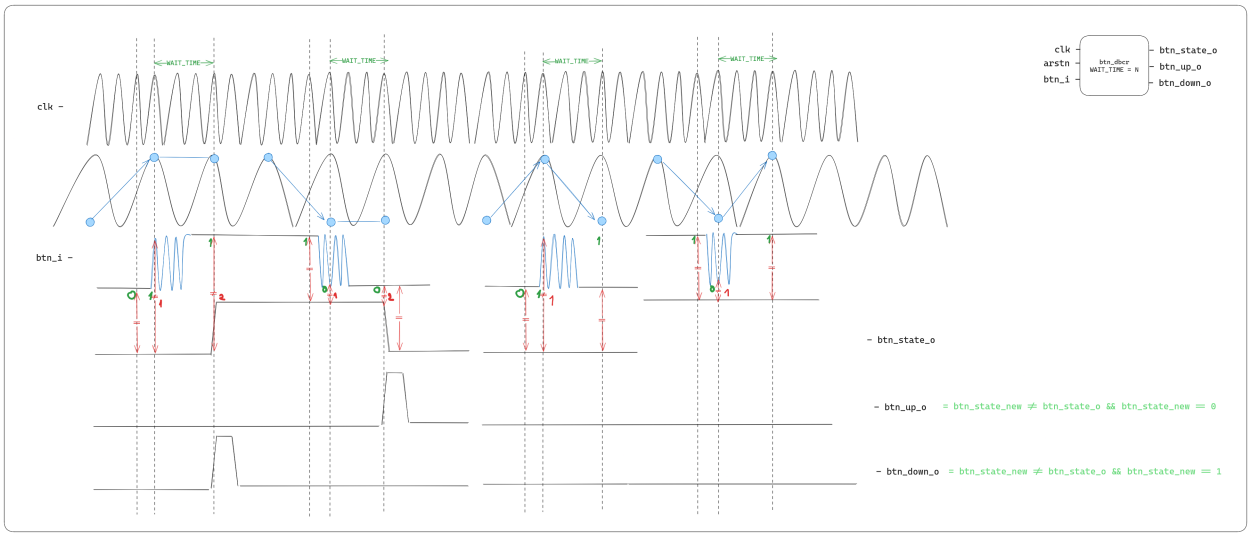


Рис. 1.2: Диаграмма сигналов модуля подавления дребезга

В процессе разработки модуля стало ясно насколько важно писать симуляционные тесты перед тем, как программировать ПЛИС и проверять на нем. Ведь компиляция для ПЛИС занимает значительно большее время и невозможно установить в точности, какие сигналы протекают внутри схемы. Потому для тестирования модуля был разработан тест симулирующий дребезг с помощью высокочастотного сигнала, рис. 1.3.

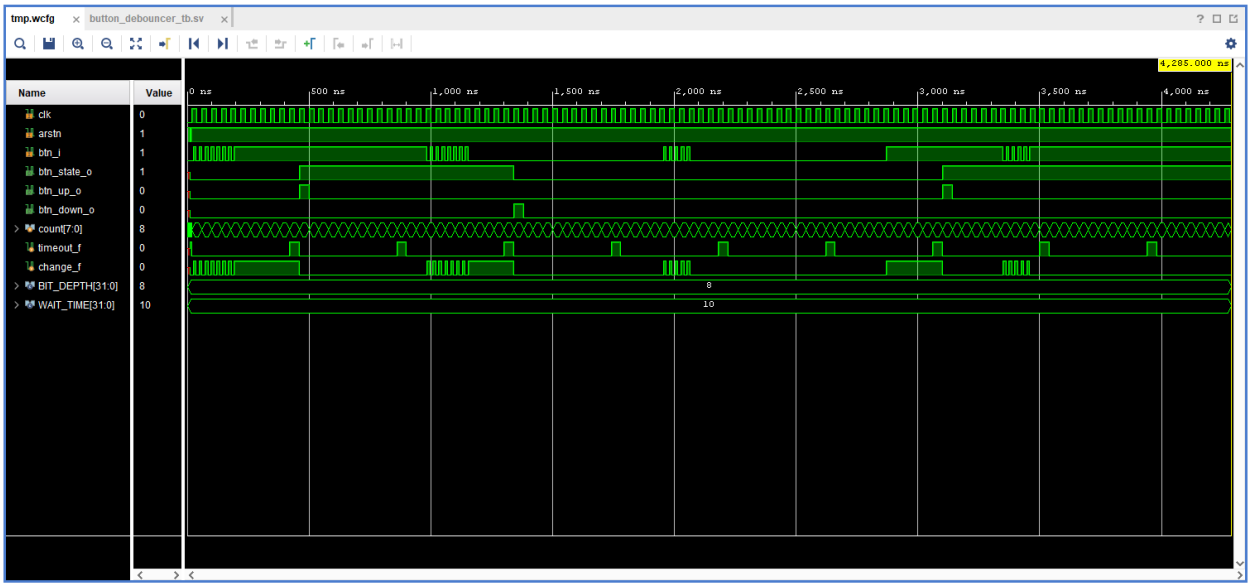


Рис. 1.3: Визуализация сигналов при работе защиты от дребезга

Возможно, разработанное мной решение не самое лучшее, но это единственное решение которое стабильно работает из других протестированных. Я рассматривал решения на основе FSM, ожидания только после обнаружения изменения сигнала и другие схемы с использованием регистров и счетчиков (о представленном в методичке способе я не знал, но после подробного его изучения, он показался не слишком подходящим).

## 3.3. Семисегментный дисплей платы ПЛИС

Управление семисегментным дисплем, на который по заданию требовалось выводить данные, оказалось неочевидным, поэтому для начала работы пришлось обратиться к официальной документации к плате и видеоурокам на YouTube («Занятие 2 (2023-24): Схемы с тактовым сигналом и состоянием. Отечественный симулятор Delta Design.» Школа синтеза цифровых схем), где наглядно объясняется принцип работы этого дисплея. Для удобной работы с ним пришлось разработать отдельный модуль, выполняющий декодирование входных сигналов в нужный набор сигналов на катоды и аноды дисплея, чтобы отображались нужные символы. Отдельную проблему представляет контроль времени, в которое нужно обновлять выходные сигналы, чтобы символы выводились в нужном порядке. Так же много проблем с отлаживанием вызвало то, что дисплей управляется активным низким уровнем сигнала. Процесс анализа ошибок усложнял тот факт, что человеческий глаз не разделяет быстро мерцающие объекты и потому иногда цифры сливались в непонятные символы.

По описанным ранее причинам, для этого модуля так же стала очевидна необходимость разработать симуляционный тест. Интересным моментом тут было то, что в устройстве драйвера имеет задержка в тысячи тактов, которые было бы трудно симулировать правильным образом, поэтому значение

задержки было вынесено во внешние параметры модуля и при тестировании задержка была по сути отключена. Временную диаграмму сигналов можно видеть на рис. 1.4.

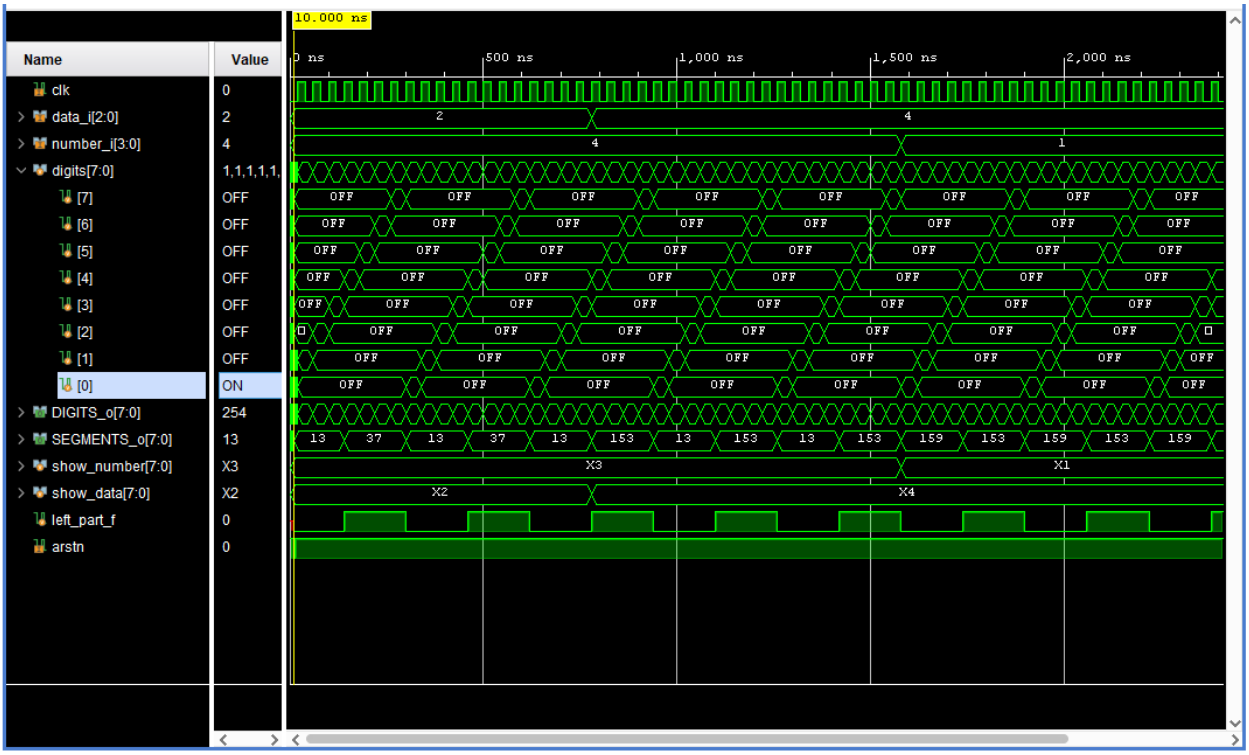


Рис. 1.4: Визуализация сигналов при работе драйвера 7-сегментного дисплея

## 3.4. Финальный результат

В результате, после отладки ошибок работы модулей, получилось синтезировать схему и запрограммировать ПЛИС на выполнение требуемой задачи. На рис. 1.5 представлен **пред**финальный результат, в финальной версии в каждой части 7-сегментного дисплея отображается только одна цифра. Система обладает интересным свойством: если при передаче данных от конкретного «транзакта» на дисплей начать их менять, то они изменятся и на дисплее, потому что арбитр не «кеширует» данные и по контракту данные на входах данные арбитра не должны меняться как только арбитр начинает их использовать. А так же у арбитра имеется один такт, в котором он перестает передавать данные и считывает новый вектор запросов для дальнейшей работы.

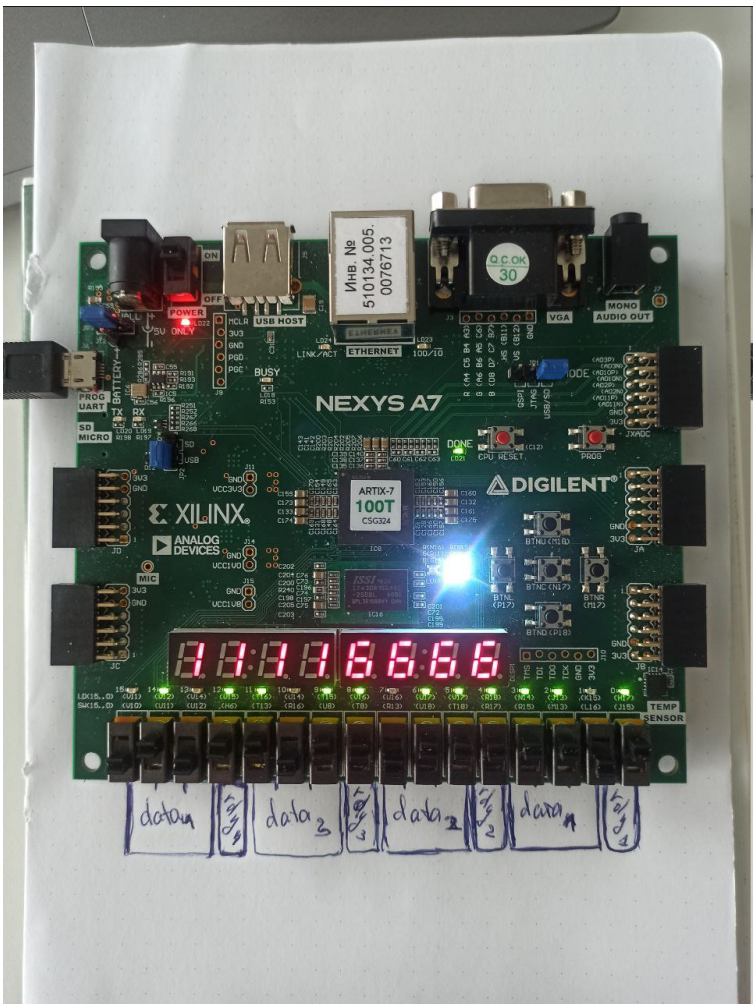


Рис. 1.5: Предфинальный результат работы разработанной схемы.

# 4. Вывод

Во время лабораторной работы хорошо чувствовалась необходимость в тестировании программ и уж тем более цифровых схем. В то время как на симуляцию тратится меньше минуты, на прошивку платы тратится несколько минут. Если не использовать возможности симуляции, процесс разработки цифровых устройств затянется очень сильно, либо в конечных изделиях будет катастрофическое количество ошибок работы.

Так же стало понятно, что разработка цифровых схем, которые выполняли бы полезную функцию, довольно сложна, но еще сложнее налаживание взаимодействия цифровых схем с внешним «физическим и биологическим миром», где имеются такие явления, как дребезг и адаптивность человеческого зрения.

Во время работы с ПЛИС обнаруживаются такие нюансы схемотехники, как пути по которым проходит тактовый сигнал: при попытке синтезировать первые версии схемы, возникала ошибка о слишком длинных путях подвода тактового сигнала, что недопустимо для качественного использования схем.

Проблема решилась добавлением флага превращающего эту ошибку в предупреждение.

В процессе разработки были выработаны новые принципы написания кода, для достижения большей выразительности, такие, например, как дополнительные сигналы разрешения и флаги. Для реализации модуля драйвера 7-сегментного дисплея было удобно использовать enum из языка SystemVerilog. И пришлось переписать модуль арбитра на более простые конструкции (без интерфейсов). Стало видно, что языки для описания аппаратуры довольно консервативные и ограниченные в своей выразительности, а результат синтеза под конкретную ПЛИС зависит от версии среды разработки, потому что в более новых версиях Vivado некоторых проблем синтеза не возникало.

В общем, получен очень ценный опыт работы с ПЛИС. Мне удалось наблюдать, что на тех же платах можно запустить целый Linux сервер, что показывает, насколько обширные возможности суть при работе с этими устройствами.