Модуль: reg\_file

Назначение:

Задача модуля заключается в реализации регистрового файла процессора.

Описание входов/выходов:

| Обозначение в коде | Назначение | Примечания |
| --- | --- | --- |
| Вход clk | тактовая частота модуля | модуль работает в нескольких доменах (домен входной шины процессора/домен user), используя для этого level\_sync для предотвращения проблем с метастабильностью |
| Вход bus | входная шина, используемая для управления модуля процессором | В симуляции информация подается не сразу с шины процесса axi3, а с упрощенной шины inter |
| Вход in | передача данных от процессора модулю в режиме записи |  |
| Вход DATATYPE | Тип входной оперируемой информации | На вход модуля по шине подаются данные, которые при помощи оператора logic определяются типами reg/wire для их обработки внутри модуля |
| Выход out | передача данных от модуля процессору в режиме чтения |  |
| Выход pulse | выходной сигнальный импульс |  |
| Выход wr | подсчет транзакций записи |  |
| Выход rd | подсчет транзакций чтения |  |

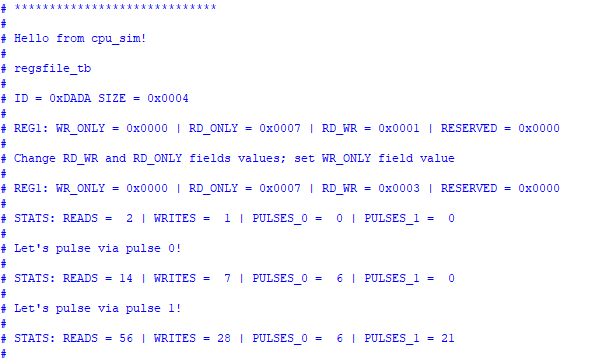
Описание параметров модуля:

| Обозначение в коде | Назначение | Примечания |
| --- | --- | --- |
| BASEADDR | адрес модуля | по умолчанию 0 |
| ID | идентификатор модуля | по умолчанию 0 |
| DATA\_WIDTH | максимальная заданнаяразрядность оперируемой информации | максимальная разрядность по умолчанию – 32 бита |
| DATATYPE | определения типа оперируемой информации | тип при помощи типа logic идет автоматическое определение типа оперируемой информации (reg/wire), исходя из ее объема |
| OUTFF | дополнительный триггер задержки | не используется |
| NREGS | подсчет регистров, используемых для хранения оперируемой информации |  |
| SYNCI | параметр выбора типа синхронизации информации в режиме записи | n – синхронизация отсутствует  l – синхронизация по уровню  d – синхронизация по данным |
| SYNCO | параметр выбора типа синхронизации информации в режиме чтения | n – синхронизация отсутствует  l – синхронизация по уровню  d – синхронизация по данным |
| INIT | параметр для инициализации работы модуля | не используется |
| NPULSE | количество сгенерированных импульсов |  |
| PULSE | массив импульсов, использующихся для работы с внешней логикой |  |
| RVALID\_FF | флаг, определяющий возможность чтения | установление rvalid означает, что информация полностью готова к чтению |

Обобщенно модуль выполняет реализацию оперируемых регистров процессора. Модуль может работать, как на чтение, так и на запись, в виде подчиненного устройства, мастером которого является процессор, связанный с регистровым файлом при помощи шины AXI3 (Advanced Extensible Interface (AXI3 or AXI v1.0)), относящейся к протоколу Advanced Microcontroller Bus Architecture (ABM) – открытому стандарту требований внутрикристалльных межсоединений для соединения и управления функциональными блоками в разработках system-on-a-chip (SoC). Использование регистров модуля может носить разнообразный характер, например, простое хранение информации или конфигурация импульсов для работы с внешними логическими системами (см. описание входов и выходов модуля).

В данном модуле применяется упрощенная шина данных процессора inter\_bus для более удобного использования (см. документацию на модуль axi3\_to\_inter).

Для начала тесты проводились **без использования синхронизации информации в режиме чтения и записи**. Для проверки корректной работы модуля был реализован подсчет транзакций записи и чтения с регистрового файла. Статистика выводится в консоли при запуске тестовой программы:

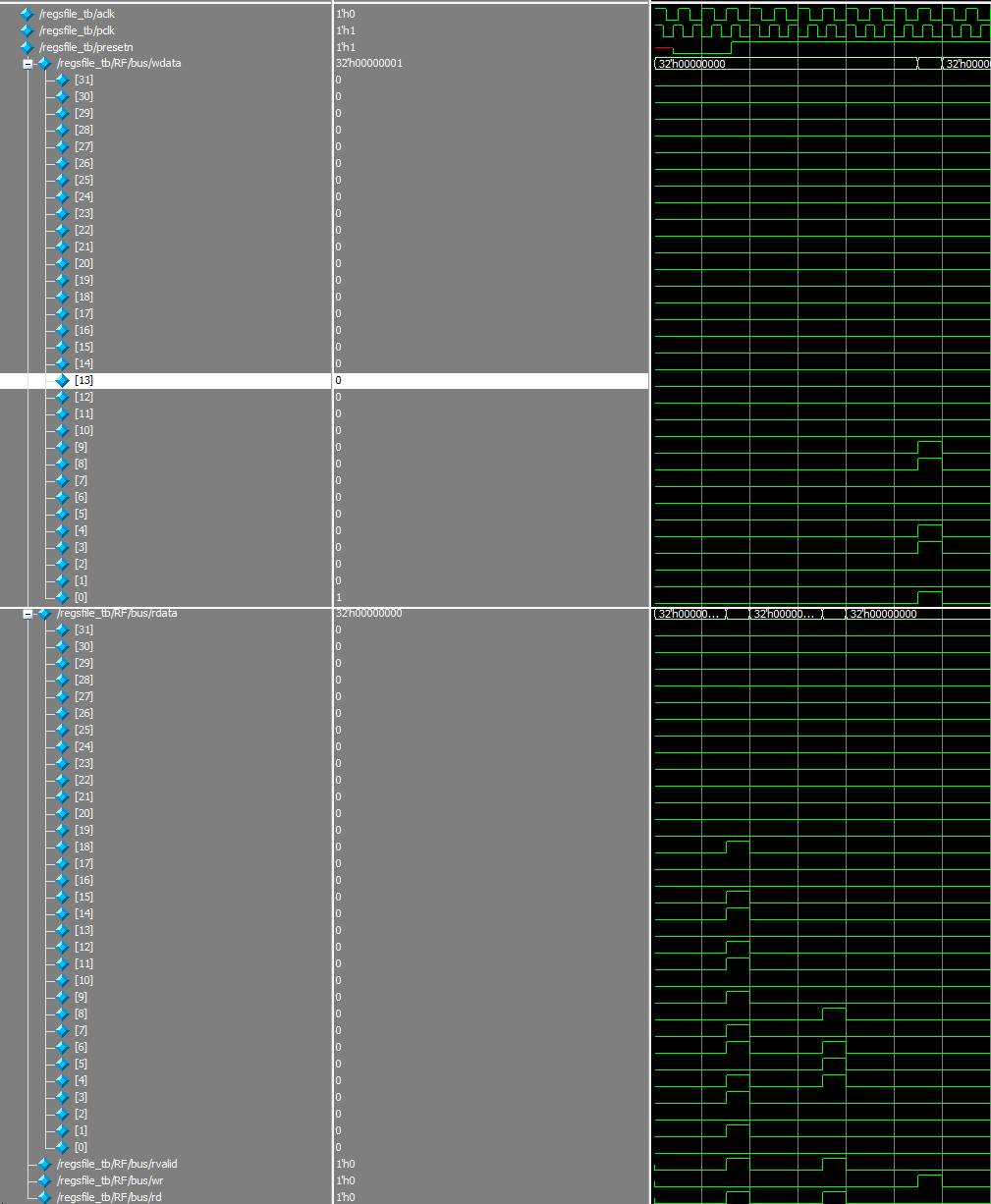


В дальнейшем при более детальном анализе полученных временных диаграмм будут присутствовать ссылки на вывод консоли выше для пояснений.

Как видно из вывода консоли, тестовая программа выводить ID регистрового файла, а также размер данных с которыми оперирует модуль. В настоящем тесте определен размер 4, что подтверждается тем, что в модуле задана разрядность оперируемой информации в 32 бита или 4 байта.

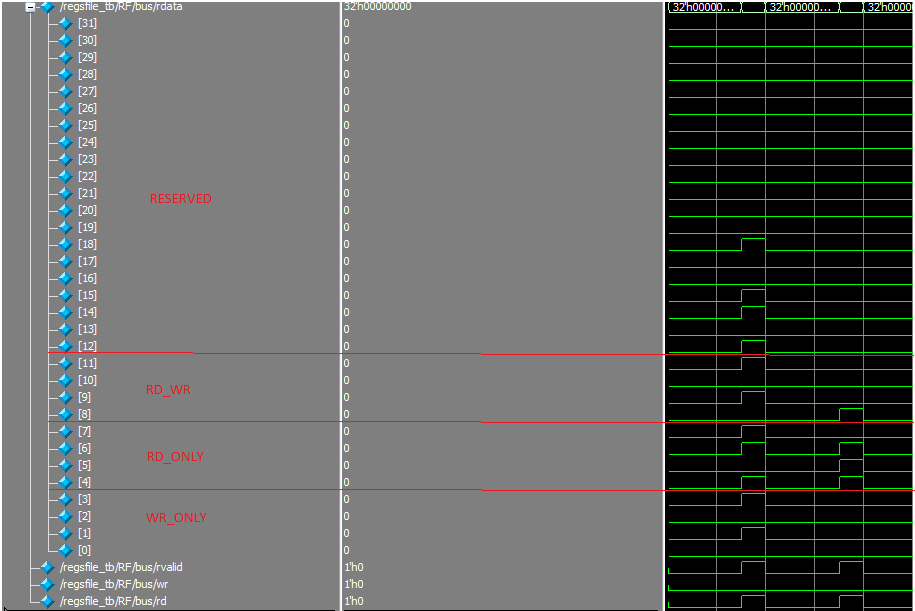
На следующей строке консоль выводит отчет по первой итерации транзакций чтения. Программно в биты RD\_ONLY и RD\_WR (только для чтения и чтение/запись соответственно) были помещены значения 7 и 1. Чтение с битов WR\_ONLY (только запись), как понятно из названия, возвращает только 0. Оставшиеся биты RESERVED (резервные биты) программно обнулены и не используются в тестах. Обычно они предназначены как буферные биты, необходимые, например, для временного хранения информации.

Далее мы проводим операцию изменения состояний битой на чтение и битов чтение/запись, а также производим запись в биты для записи. Программно мы прибавляем к значению битов чтения/записи 2, записываем в биты записи число 9, а также пытаемся записать в биты чтения 1. Видно, что статистика изменилась только в состоянии битов чтение/запись. Причиной этому является то, что биты на чтения не могут подлежать записи, как и биты на запись не могут подлежать прочтению. В первом случае состояние битов никак не изменяется, во втором же происходит запись, но прочитать биты, чтобы это увидеть мы не можем. На самом деле запись действительно происходит, что видно на диаграммах ниже:



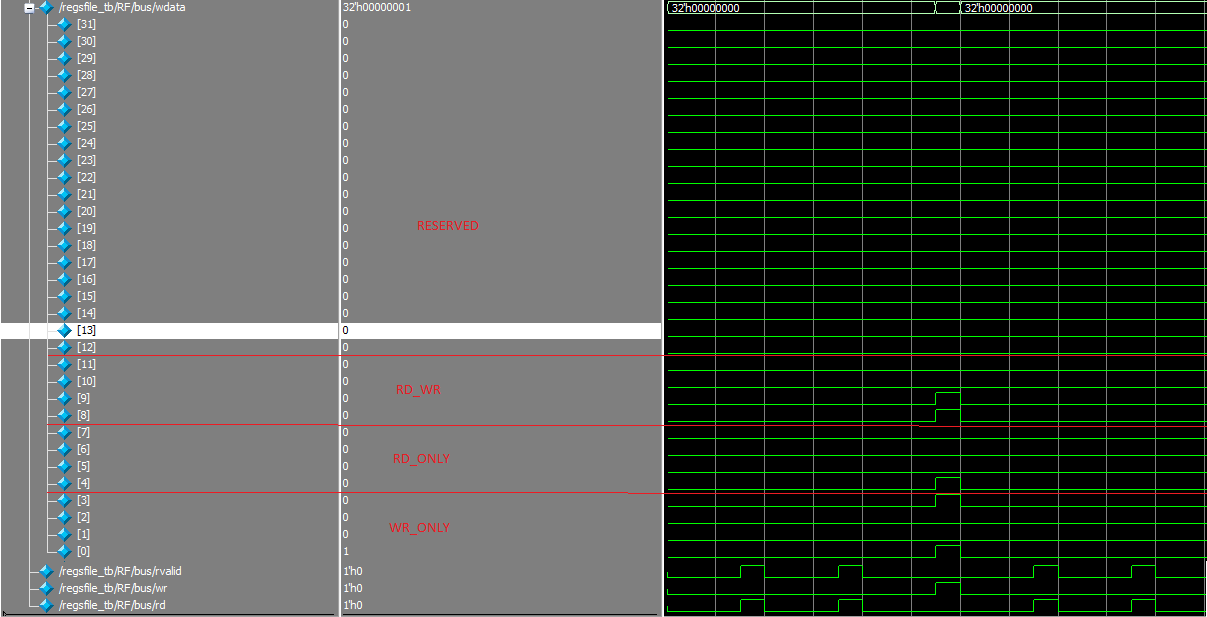
Здесь приведены состояния тактовых сигналов aclk и pclk, сбросового сигнала presetn, состояния всех битов шины rdata, то есть чтения, состояние битов шины wdata, сиречь данный на запись, флагового сигнала rvalid и счетные сигналы wr и rd, которые, как писалось ранее подсчитывают количество транзакций записи и чтения соответственно. Разберем по порядку:

1. Тактирующие сигналы существуют в двух доменах , оба из которых приведены в диаграмме. Как видно, все изменения происходят по фронту pclk – тактовой частоты процессорной шины.
2. Сбросовый сигнал определяет начало работы модуля. По его положительному перепаду происходит первая транзакция чтения
3. Данные на чтения приведены в виде 32 разрядной шины, поделенную на сегменты различных битов. Так в тестовом модуле по 4 бита уделяется битам записи, чтения и чтения/записи, а оставшиеся 20 битов являются резервными. выделим операции чтения и разберем их более подробно, выделив сегменты различных битов:

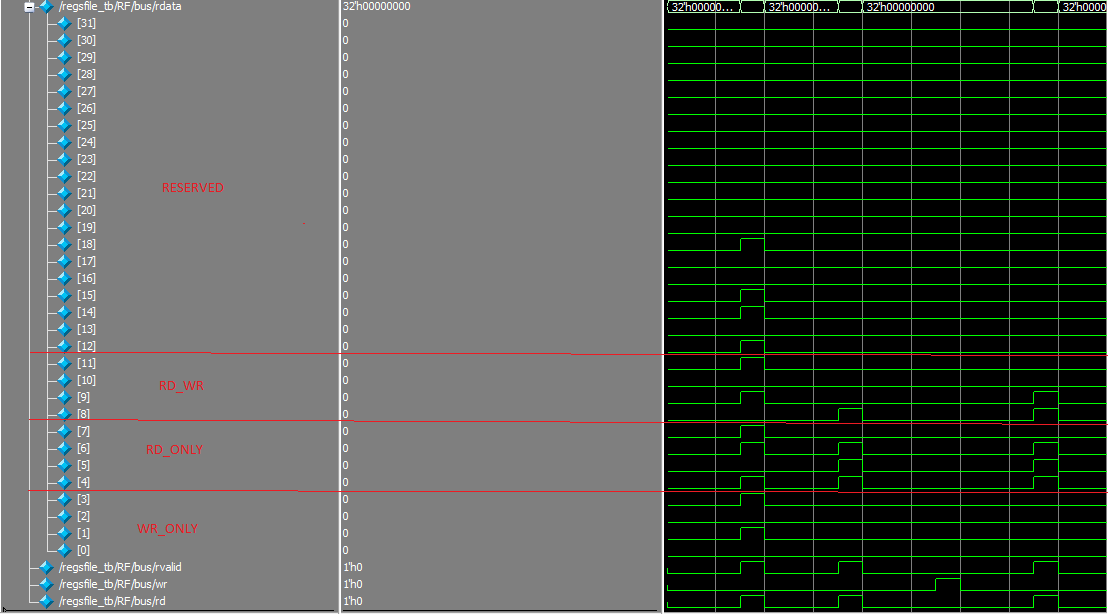


Первое изменение шины чтения происходит из-за инициализации модуля по фронту сигнала presetn при этом видно, что участие принимают и зарезервированные биты. Второе же изменение шины показывает то, что мы и видели в консоли. Биты на запись пусты, как и предполагалось, на битах чтения комбинация 0111, что при переводе из двоичной системы равняется 7, а на битах чтение/запись – 0001, что соответственно равняется 1. Также мы видим, что сигнал rvalid показывает 1 в одно время с итерациями чтения. Это показывает, что поднимается условный флаг готовности данных к чтению. На сигнале rd также наблюдаются единицы в моменты чтения, что используется в подсчете количества соответственных итераций.

1. Данные на запись аналогично представлены 32 битной шиной. После двух итераций чтения, рассмотренных выше, происходит запись, рассмотрим ее подробнее:

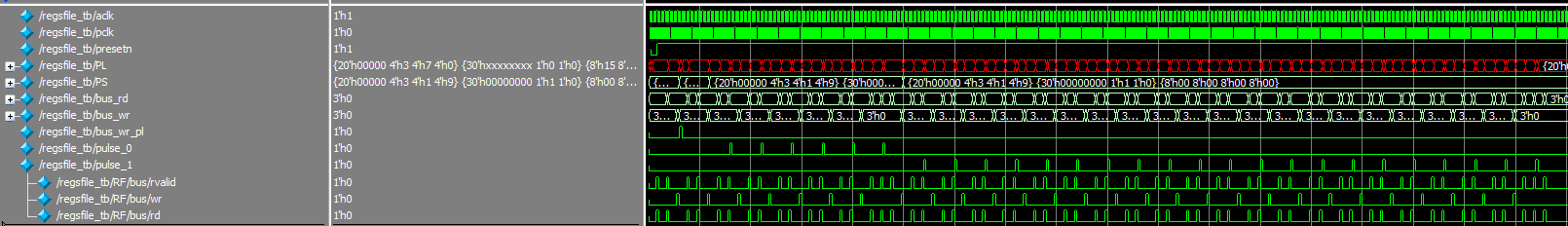


Как видно из диаграмм происходят операции описанные выше: в биты записи пишем 9, пытаемся писать в биты чтения, прибавляем к значению битов чтения/записи 2, то есть из 1 получаем 3. Также наблюдаем единицу на сигнале wr, что используется для счета итераций записи. Посмотрим, что нам покажет чтение битов:

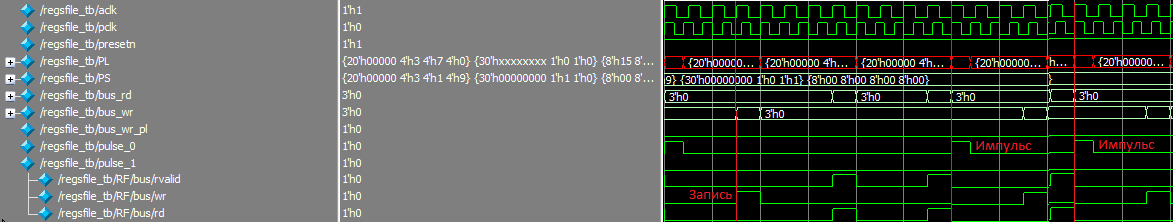


Как видно на последней итерации чтения, в битах записи мы все также ничего не наблюдаем, в битах чтения изменений нет, а биты чтения/записи изменились в соответствии ожиданиям.

Далее рассмотрим возможность модуля генерировать импульсы. Так мы имеем два бита, отвечающие за импульсы. Программно зададим модулю генерацию 6 импульсов на одном бите и 21 на другом. На временных диаграммах получаем:



Как мы видим нужное количество импульсов было сконфигурировано. Каждый импульс требует двух итераций чтения и одной итерации записи. К тому же видна задержка между записью и появлением импульса:



Поскольку запись и импульс происходят в разных тактовых доменах, задержку следует обозначить в тактах pclk, то есть 11 тактов.

**Проверка работы параметров синхронизации.** Синхронизация информации реализована в нескольких вариациях (см. описание параметров модуля). Так синхронизация может идти по уровню либо по данным и реализованы она через соответствующие модули level\_sync и data\_sync (см. документацию на level\_sync и data\_sync). В обоих этих модулях реализована схема борьбы с метастабильностью (неопределенным состаянием сигнала при его реализации в нескольких частотных доменах, см. пояснительный рисунок ниже).

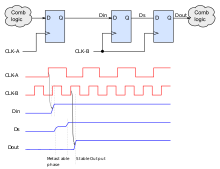
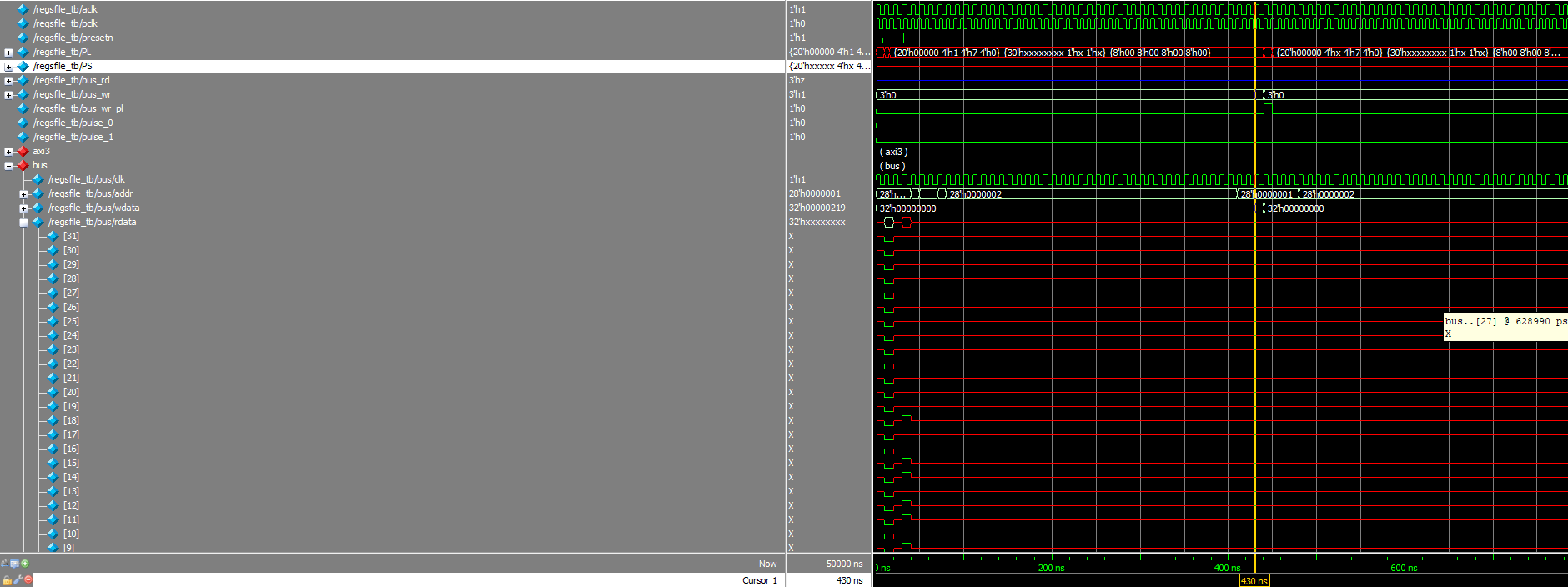


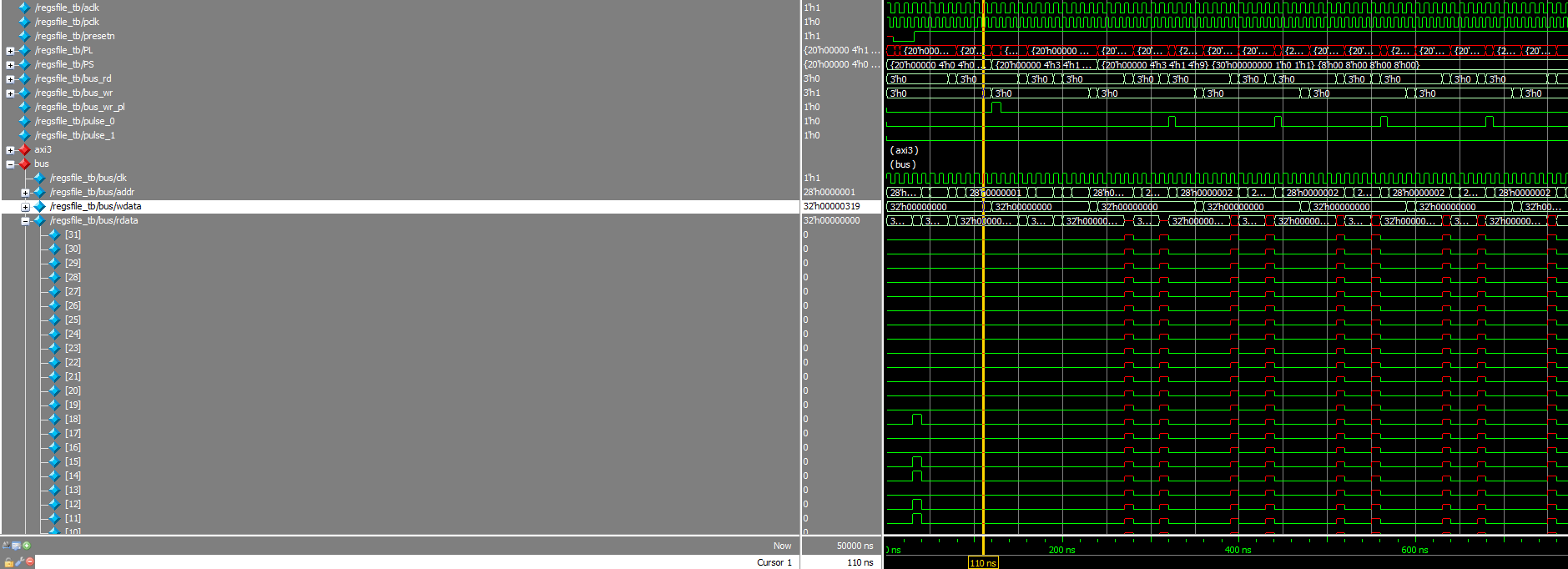
Схема представляет собой линию задержки, реализованную через триггер задержки. В теории, использование любого из приведенных выше типов задержки будут наблюдаться задержки в выводе информации на чтение и запись на 1 или более тактов.

**Проверка режимов синхронизации:**

1. **Оба параметра синхронизации в положении “d” – синхронизация по данным:**

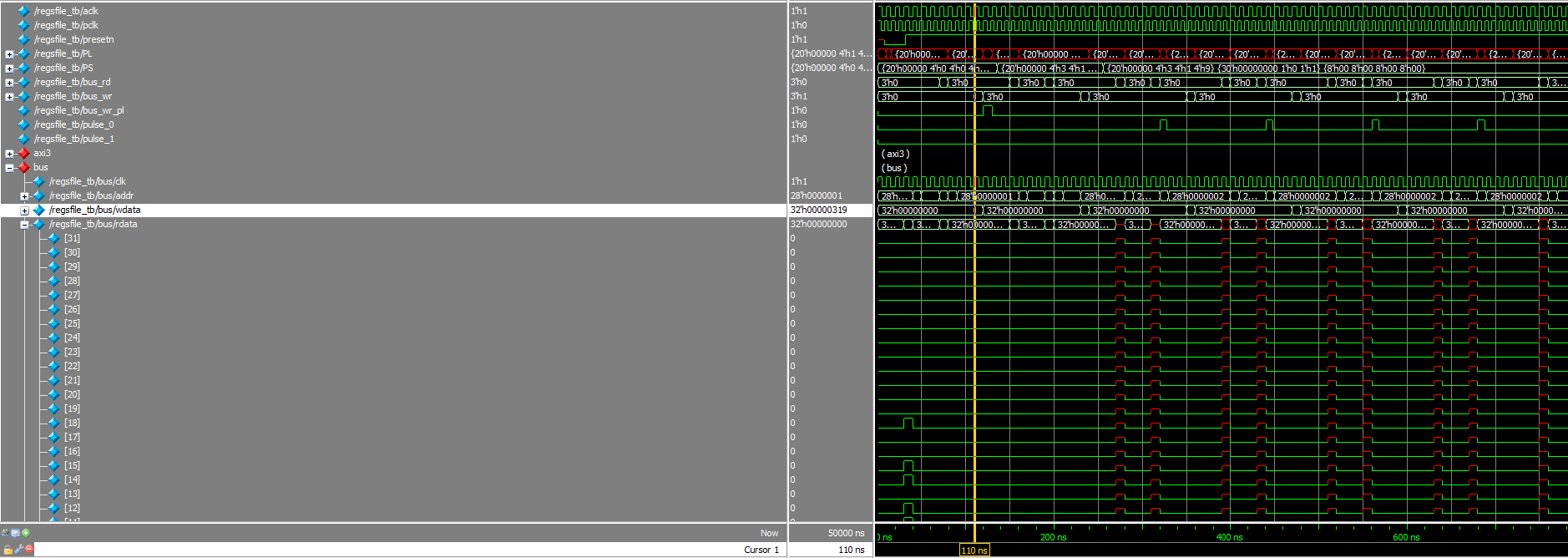
****

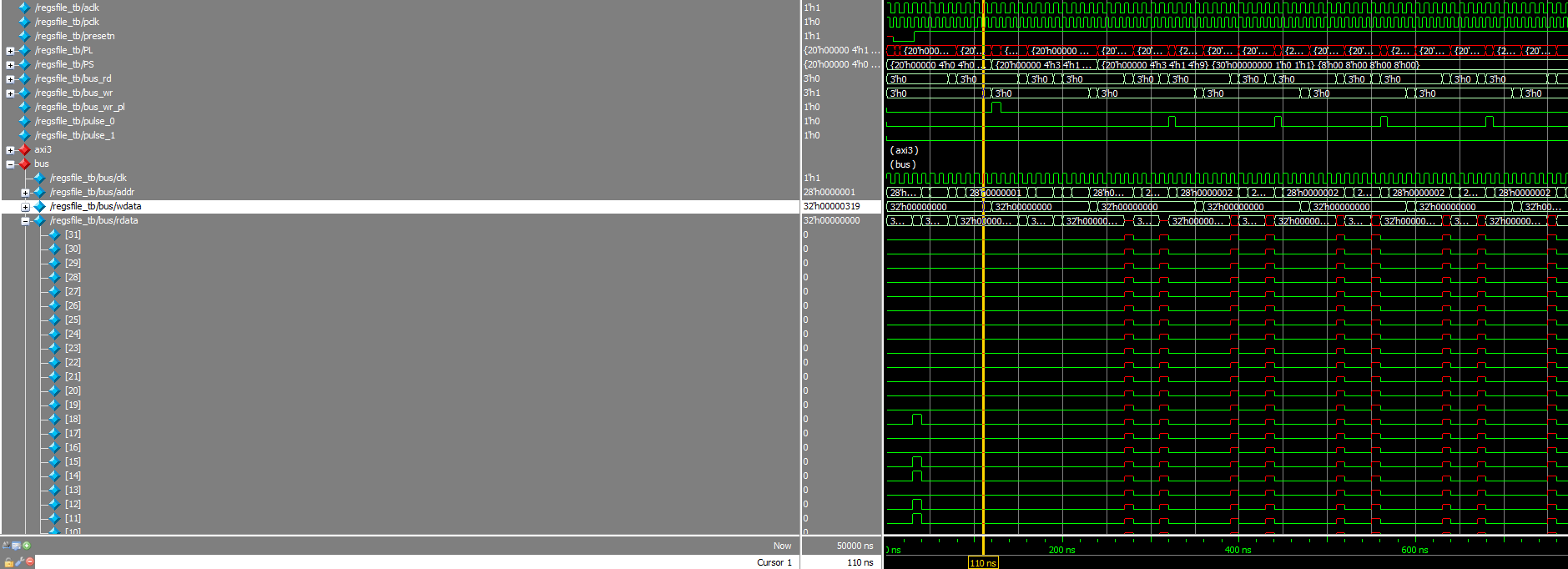
На диаграмме видно, что при использовании такого режима синхронизации, данные на чтение с регистрового файла не передаются корректно. Желтой линией отмечен момент записи в регистровый файл. Сравним задержки без синхронизации:



Без синхронизации чтение происходит без ошибок.

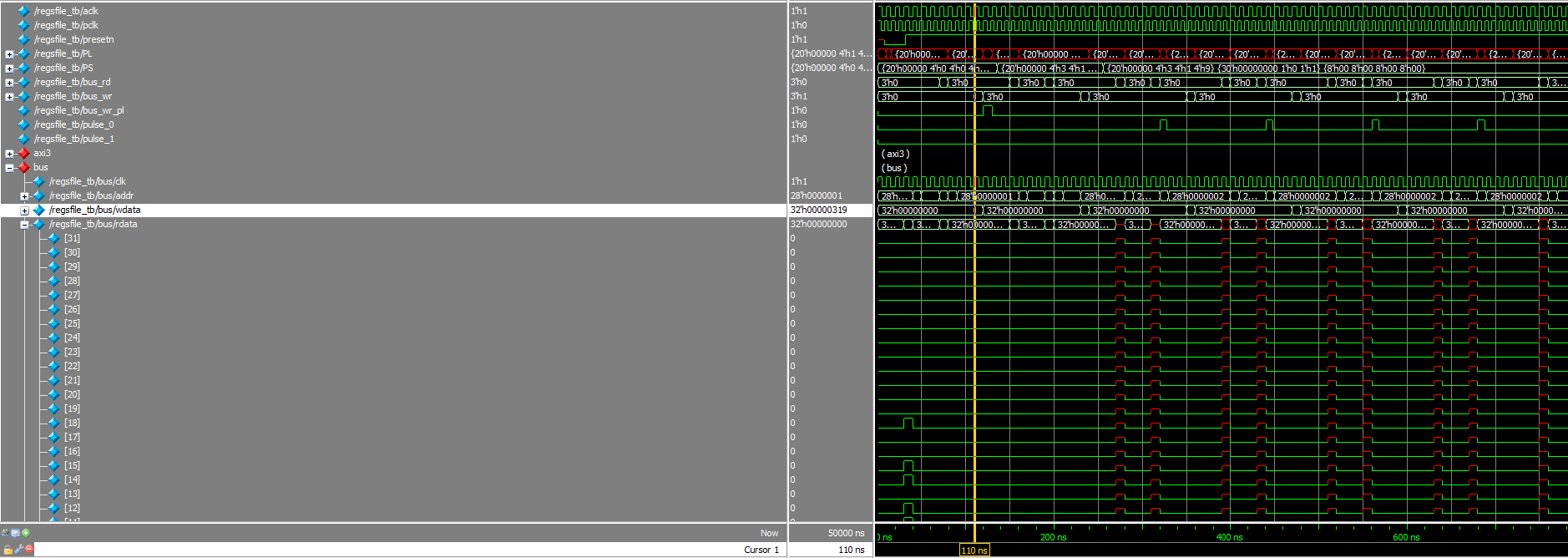
1. **Оба параметра синхронизации в положении “l” – синхронизация по уровням**

****Из диаграммы видно, что она полностью соответствует оной без синхронизации (см. ниже).



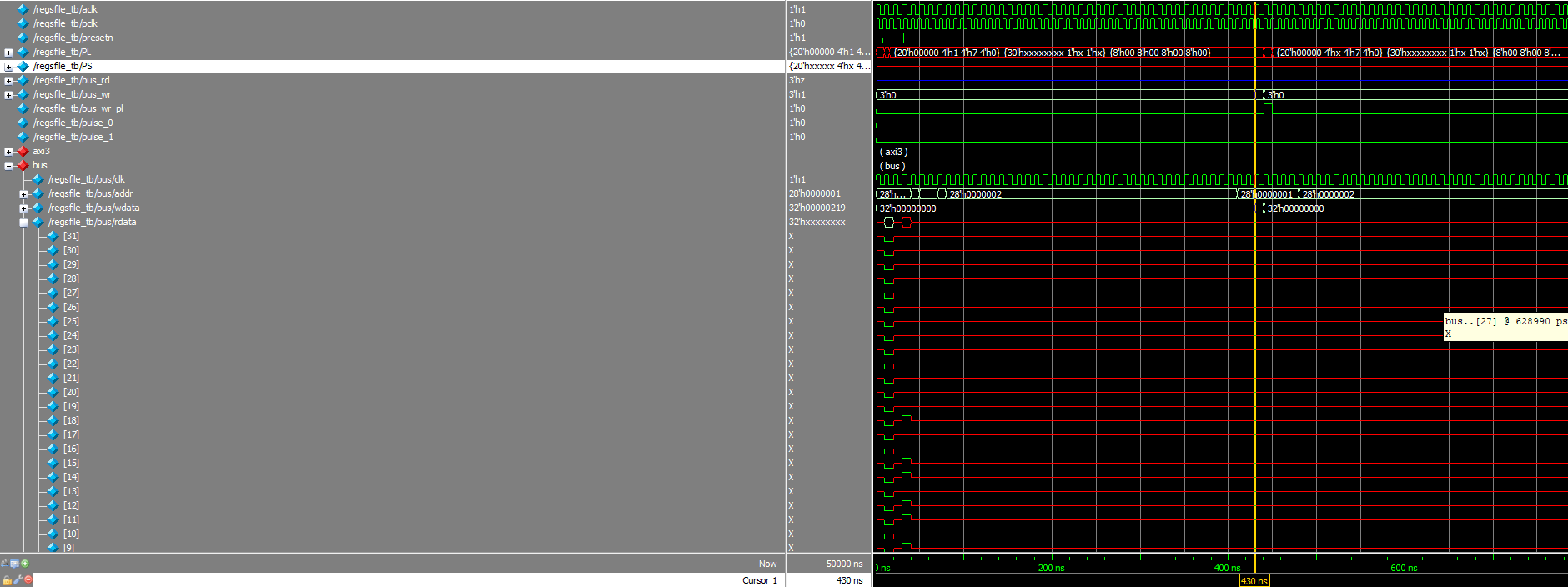
1. **SYNCI – l, SYNCO – d**

Без изменений:

****

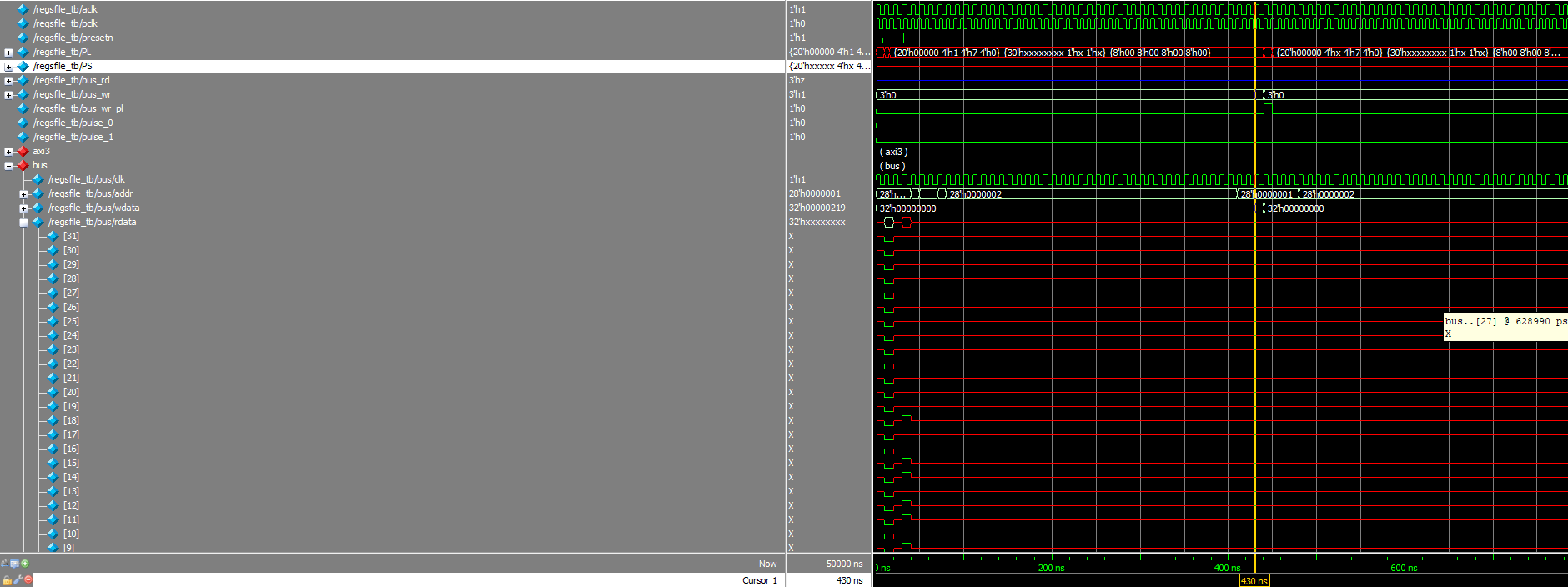
1. **SYNCI – d, SYNCO – l**

Полностью соответствует картине из пункта 1:

****

1. **SYNCI – d, SYNCO – n**

Полностью соответствует картине из пункта 1:

****

Из тестов выше видно, что картины временных диаграмм не зависят от значения параметра SYNCO, так как результат одинаков вне зависимости от режима.Изменения наблюдаются только при изменении параметра SYNCI на режим синхронизации по данным: наблюдаются существенная задержка в режиме записи и полная потеря работоспособности в режиме чтения. Положение синхронизации по уровням в параметре SYNCI никакого эффекта не имеет.