АННОТАЦИЯ

Суть данной курсовой работы заключается в анализе проблематики разработки мультисинхронных цифровых устройств, разработке и реализации устройства для ресинхронизации данных.

Работа состоит из пояснительной записки и графической части. Графическая часть представляет собой описание реализуемого программного решения.

Пояснительная записка содержит 3 раздела. В первом разделе освещается проблематика создания мультисинхронных цифровых устройств, то есть устройств, блоки которых используют более чем один синхросигнал, а также пути организации взаимной работы клоковых доменов. Второй раздел посвящен проектированию разрабатываемого цифрового устройства по ресинхронизации данных. Третий раздел посвящен процессу реализации цифрового устройства на языке описания аппаратуры Verilog на основании спроектированных схем и моделей.

Работу выполнил студент группы ИВБО-02-19 Денисов К.Ю.

Руководитель — Тарасов И.Е.

Работа содержит 33 страницы, 13 рисунков, 4 таблицы, 2 приложения. В работе использованы 0 источников.

Первое приложение содержит в себе исходные коды модулей на языке Verilog, разрабатываемых в ходе курсовой работы.

Второе приложение содержит результаты симуляции цифрового устройства в САПР ISE Design Suite.

СОДЕРЖАНИЕ

ПЕРЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

CDC — Clock domain crossing

FIFO — First in, first out

FPGA — Field-Programmable Gate Array

MSB — Most significant bit

ВВЕДЕНИЕ

Целью данной курсовой работы является построение цифрового устройства ресинхронизации данных. Данное устройство должно реализовывать способ организации данных «first in, first out», иметь восьмиразрядный вход и тридцатидвухразрядный выход.

Данные на выход должны подаваться по мере образования во входном буфере не менее четырех восьмиразрядных слов. Тактовые сигналы входного и выходного каналов независимы, однако тактовая частота для выхода устройства составляет как минимум четверть от тактовой частоты входного интерфейса устройства.

Актуальность данной работы состоит в том, что большинство современных устройств, работающих с данными, включая микроконтроллеры, модемы, сетевые интерфейсы и сетевые процессоры, имеет множество последовательных интерфейсов и различных сигналов, которые производят передачу данных и сигналов управления через многие домены синхрочастоты.

Все разрабатываемые модули необходимо спроектировать с использованием языка описания аппаратуры Verilog.

Проектирование, разработку и тестирование модулей реализуемых устройств планируется осуществлять средствами ISim в САПР ISE Design Suite.

Исходные данные к курсовой работе включают входные данные для записи в очередь FIFO.

1 ПРОБЛЕМАТИКА СИНХРОНИЗАЦИИ

В данном разделе будут рассмотрены проблемы и особенности построение устройств

В этом разделе будут описаны проблемы, возникающие в процессе разработки мультисихронного проекта, то есть устройства, в котором имеют место пересечения клоковых доменов или доменов синхрочастоты (CDC).

1.1 Домен синхрочастоты

Домен синхрочастоты представляет собой ту часть проекта, которая тактируется одной или несколькими синхрочастотами, причем все эти синхрочастоты должны иметь постоянные сдвиги фазы. Если в какой-либо части проекта имеется синхрочастота или инвертированная синхрочастота, или синхрочастота, полученная из исходной путем деления на 2, то такая часть проекта считается клоковым доменом с одной синхрочастотой. Если же домены имеют синхрочастоты с переменной фазой и соотношениями времени, то такие домены считают доменами с различными синхрочастотами.

На Рисунке 1.1 показано, что проект имеет единственный домен синхрочастоты, потому что синхрочастота divClk — есть деленная на два частота генератора синхронизации Clk.

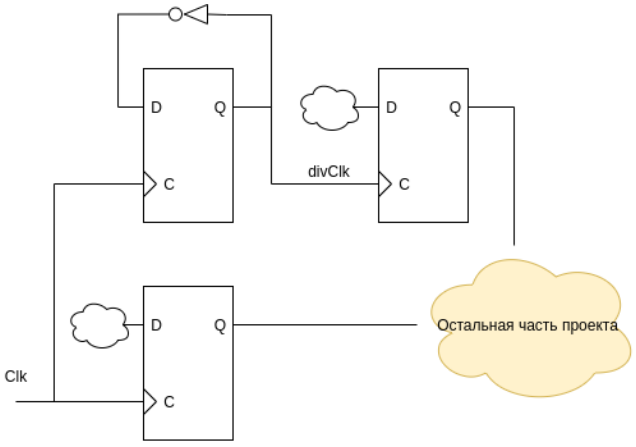

Рисунок 1.1 — Схема с одним доменом синхрочастоты

На Рисунке 1.2 показано несколько синхрочастот от различных источников. Ту часть проекта, которая управляется этими синхрочастотами, называют доменами синхрочастоты, и сигналы, осуществляющие передачу импульсов между этими асинхронными доменами синхрочастоты, называют путями пересечения домена синхрочастоты. Сигнал DA считают асинхронным сигналом в домене синхрочастоты, так как между генератором синхронизации A (clkA) и генератором синхронизации B (ClkB) не существуют постоянные соотношения фазы и времени.

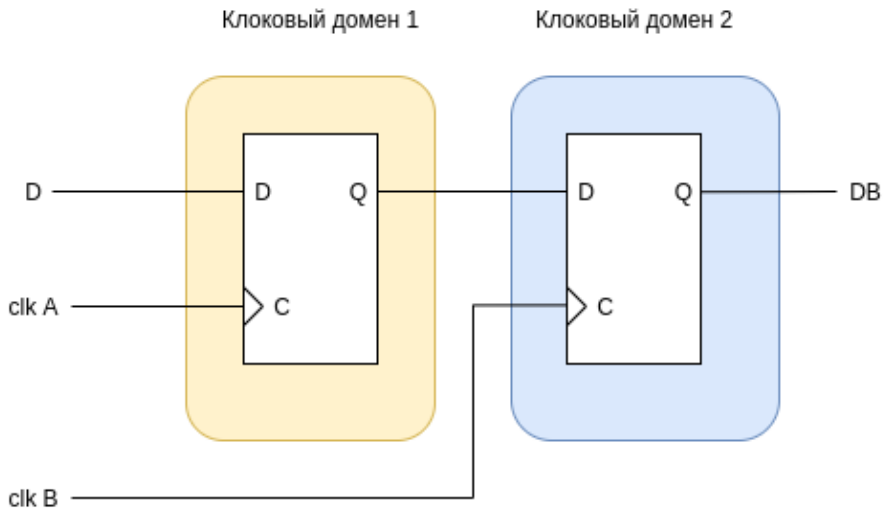

Рисунок 1.2 — Путь домена синхрочастоты

1.2 Основные принципы

При разработке ультисинхрочастотных проектов следует уделять особое внимание стабильности сигнала. Когда сигнал пересекает домен синхрочастоты, то он появляется в новом домене синхрочастоты как асинхронный сигнал и должен быть засинхронизирован.

Синхронизация предотвращает в новом домене синхрочастоты метастабильное состояние первого запоминающего элемента схемы (триггера), и это позволяет в новом домене работать уже со стабильным сигналом.

Метастабильность — это неспособность триггера достигнуть известного состояния в определенный момент времени. Когда триггер входит в метастабильное состояние, то невозможно предсказать ни уровень выходного

напряжения элемента, ни период времени, за который этот выход перейдет к правильному уровню напряжения.

В течение этого переходного времени выход триггера будет находиться на некотором промежуточном уровне напряжения или колебаться и может передать этот недопустимый уровень сигнала со своего выхода к другим триггерам схемы.

1.3 Решение проблемы метастабильности

С целью решения проблемы метастабильности применяются каскады стабилизирующих триггеров, включенных последовательно. Рассмотрим это решение.

Простейший синхронизатор представляет собой два триггера, включенных последовательно без какой-либо комбинационной схемы между ними. Такая схема проекта гарантирует, что первый триггер выходит из своего метастабильного состояния, и его выход переходит в устойчивое состояние перед тем, как второй триггер сохраняет его. Необходимо также разместить эти триггеры, насколько возможно, ближе друг к другу для того, чтобы гарантировать наименьшую расфазировку тактовых сигналов между ними.

Другой тип ячейки синхронизатора представляет собой два близко расположенных триггера без какой-либо комбинационной логики между ними. Для того чтобы синхронизация работала должным образом, сигнал, пересекающий домен синхрочастоты, должен проследовать от триггера в домене синхрочастоты источника сигнала к первому триггеру синхронизатора, не пройдя через комбинационную логику.

Синхронизацию в домене называют привязкой входного сигнала к тактовой частоте домена, т. е. все изменения этого сигнала в домене будут происходить по фронту или срезу тактового сигнала домена, а не родительского тактового сигнала. На Рисунке 1.3) приведена схема традиционного синхронизатора из двух триггеров (сдвиговый регистр) для одиночного сигнала. Схема синхронизации приведена на Рисунке 1.4.

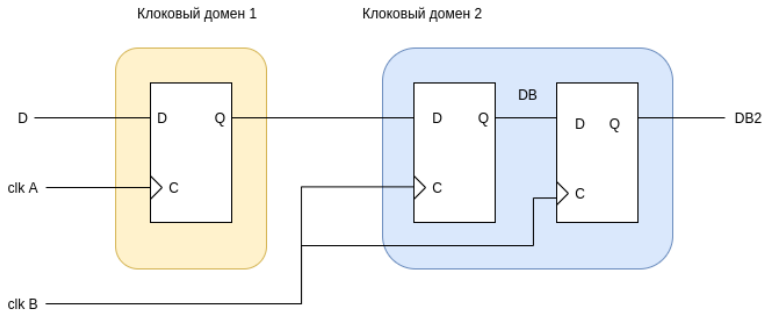

Рисунок 1.3 — Триггеры синхронизации

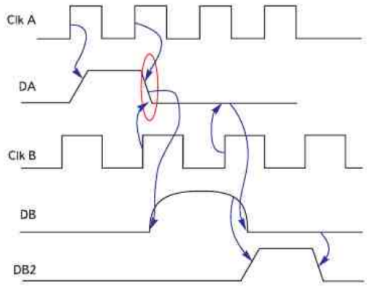

Рисунок 1.4 — Схема синхронизации сигнала

Данный способ борьбы с метастабильностью будет в дальнейшем использован при разработке узла ресихронизации данных.

1.4 Асинхронная очередь

Во многих проектах необходимо передавать из одного домена в другой не только несколько одиночных сигналов, но и сигналы типа шин: шины данных, адреса и шины управления.

В этих задачах активно используются буферы типа FIFO и специальные протоколы для процедуры установления связи. Особенно полезной оказывается асинхронная очередь, позволяющая производить запись и чтение по синхросигналам двух клоковых доменов. Данная схема позволяет быстро и удобно передавать данные в реальном времени между двумя разными областями синхронизации.

На Рисунке 1.5 приведена схема использования двухпорторой оперативной памяти, позволяющей осуществлять передачу данных между частями устройства, использующими синхросигналы разной частоты.

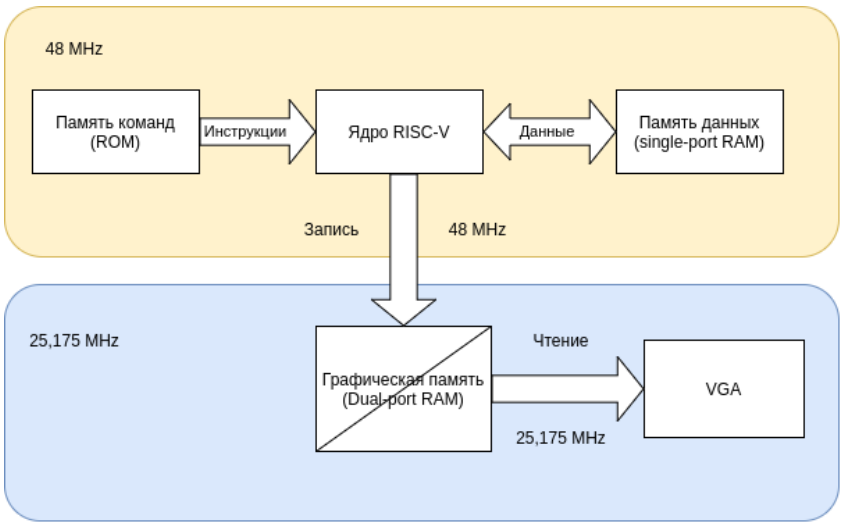

Рисунок 1.5 — Двухпортовая оперативная память

Двухпортовая память с независимыми тактовыми сигналами — аппаратный блок, используемый в современных FPGA и в технологических библиотеках для заказных микросхем. Чтение и запись производится совершенно независимо через два отдельных порта.

Единственным ограничением является одновременное обращение на запись и чтение по одному и тому же адресу памяти — оно может привести к неопределенному результату. На основе такого блока памяти зачастую создается модуль FIFO, который позволяет с одной стороны записывать данные из одного тактового домена, а с другой — забирать в другой тактовый домен. Заодно логика FIFO следит за тем, чтобы не происходило обращения к одной и той же ячейке памяти.

Данный узел работает по следующему принципу.

1. Источник данных загружает данные в асинхронную очередь. Тактирование производится по тактовому сигналу источника данных.
2. Потребитель данных считывает данные из асинхронной очереди в исходном порядке. Тактирование производится по тактовому сигналу потребителя данных.

2 ПРОЕКТИРОВАНИЕ ЦИФРОВОГО УСТРОЙСТВА

В данном разделе освещается разработка функциональной и структурной схем устройства по ресинхронизации данных. Определим структурные элементы устройства, принцип и порядок его работы.

2.1 Организация взаимодействия с устройством

Взаимодействие с устройством производится посредством передачи восьмибитного слова на вход wdata. Аппаратный блок осуществляет запись данных во внутреннюю память по сигналам тактового генератора домена источника данных wclk, основываясь на состоянии входных сигналов wrst, wen и служебных сигналов wr_full, wptr, rptr.

Чтение из устройства ресинхронизации производится посредством считывания тридцатидвухразрядного слова, составляемого из четырех последовательно поданных на вход восьмиразрядных слов в порядке *от старшего к младшему* (англ. big-endian), генерируемого на выходе rdata. Аппаратный блок осуществляет чтение данных из внутренней памяти по сигналам тактового генератора домена получателя данных, основываясь на состоянии входных сигналов rrst, ren r_empty wptr, rptr, rdredy.

2.2 Организация памяти

Устройство хранит данные в собственной памяти. Устройство может одновременно хранить шестнадцать восьмибитных слов, используя для их адресации пятиразрядный адрес, в котором старший бит призван реализовать возможность контроля заполнения и опустошения очереди.

Возможность асинхронного сброса памяти не реализована в силу того, что обычно память в устройствах, реализуемых на ПЛИС представлена в виде

блочной оперативной памяти и выделенной оперативной памяти. Эти ресурсы обычно не поддерживают сброс. Сброс возможно реализовать в регистрах.

2.3 Тактирование

Управление устройством производится по двух синхросигналам wclk и rclk. С целью оптимальной работы цифрового устройства при передаче данных из одного домена синхросигнала в другой тактовые сигналы входного и выходного каналов сделаны независимыми, однако тактовая частота для выхода очереди составляет как минимум $\frac{1}{4}$ от тактовой частоты входного интерфейса FIFO.

Для корректной передачи и предотвращения потери данных, а также с целью борьбы с метастабильностью в случае независимых тактовых сигналов используются группа синхронизирующих триггеров, которые передают текущее состояние адреса чтения и адреса записи между клоковыми доменами. Основываясь на значениях данных адресов делается вывод о заполненности и пустоте очереди.

2.4 Адресация

Адресация в памяти реализована с помощью двух пятиразрядных переменных, хранящих адрес ячеек, с которыми будут произведены операции чтения и записи при следующем синхросигнале.

Важной частью работы устройства является работа с памятью. Необходимо избежать потери и перезаписи не переданных данных. С этой целью на каждом шаге работы схемы осуществляется проверка и сравнивание адресов чтения и записи.

Во время работы схема хранит адреса в виде рефлексивного двоичного кода Грея. Выбор в сторону кода Грея сделан для снижения критичности ошибки определения адреса чтения или записи при его передаче.

В случае использования кода Грея при переходе к следующему адресу изменяется только один бит, и даже если схема не сможет верно определить значение адреса — посчитает, что адреса не изменился, это не приведет к нарушению работы системы.

В то время как при использовании прямого двоичного кода ошибка синхронизации неизбежна. При переходе могут изменяться все биты адреса (например при переходе $01111 \rightarrow 1000$). В данном случае значение адреса является полностью неопределенным. Это может привести к нарушению в работе устройства в целом.

Когда указатели чтения и записи равны, это указывает на то, что FIFO пуст. Эта ситуация возникает во время операции сброса или когда было произведено чтение последнего слова в FIFO, как показано на Рисунке 2.1.

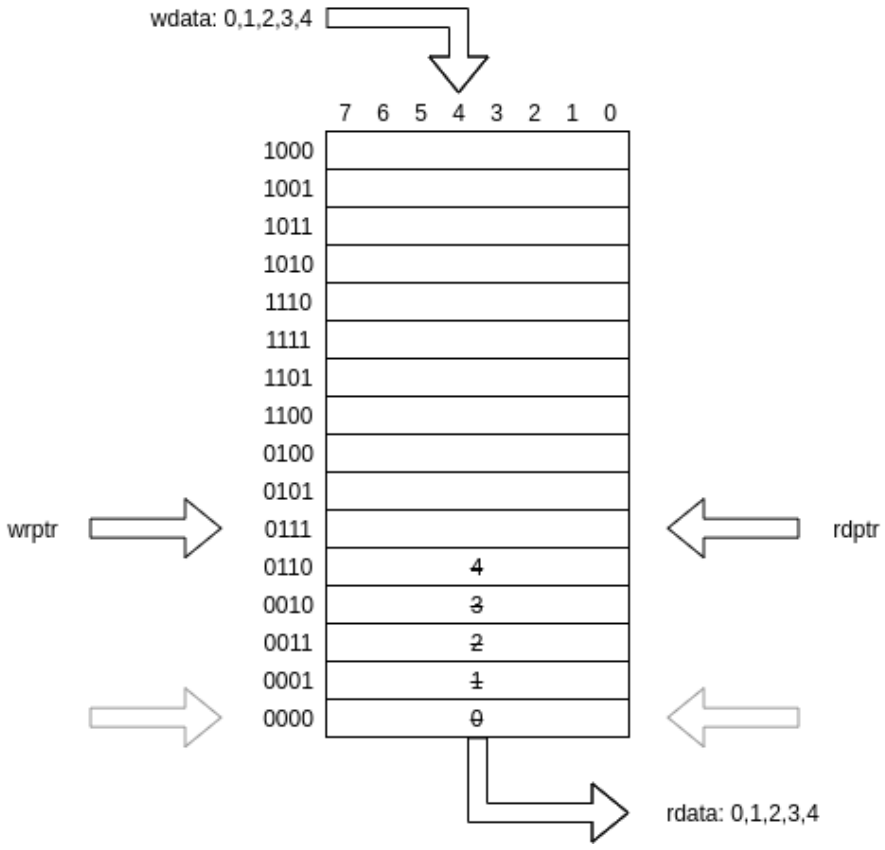

Рисунок 2.1 — Все данные прочитаны. Очередь пуста

Когда указатели чтения и записи снова равны, это означает, что очередь заполнена и дальнейшая запись не может быть произведена. Эта ситуация возникает, когда переменная адреса записи переполняется, как показано на Рисунке 2.2.

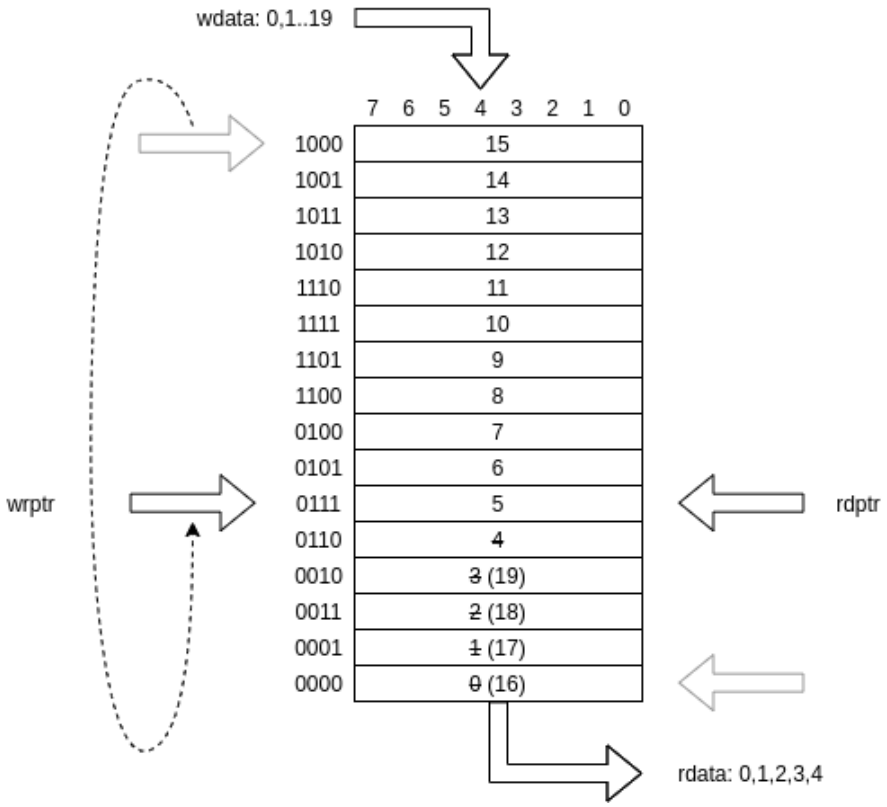

Рисунок 2.2 — Очередь заполнена. Запись не допустима

В обоих случаях адреса записи и чтения указывают на одну и ту же ячейку памяти, но при этом ситуации являются противоположными, что усложняет логику определения текущего состояния памяти цифрового устройства.

С целью определения текущего состояния асинхронной очереди необходимо использовать на один бит больше, чем требуется для адресации всех ячеек запоминающего устройства. Сравнение этого бита даст ответ о состоянии памяти в случае, когда указатели чтения и записи окажутся равны.

Пятиразрядные сигналы wr_full и rd_empty, позволяющие организовать корректную работу с устройством, формируются с помощью следующих логических условий.

$$wr\_full = (wptr == \{!rptr[4:3], rptr[aw-2:0]\})\,; \tag{2.1}$$

$$rd\_empty = (wptr == rptr) \tag{2.2}$$

В устройстве необходимо реализовать проверку обращения к одному элементу памяти. Чтение по данному адресу должно быть заблокировано до тех пор, пока адрес записи не перестанет указывать на данную ячейку памяти.

При каждом приходящем тактовом сигнале устройство осуществляет сравнение адресов чтения и записи, предварительно производя с ними промежуточные действия. Приведем список преобразований и производимых операций.

— перевод из прямого двоичного кода в рефлексивный код Грея;

— сохранение текущего значения в регистре;

— передача значения адреса в синхронизационные регистры;

— перевод из рефлексивного кода Грея в прямой двоичный код;

— синхронизация приходящего адреса генератором соседнего домена синхронизации;

— сравнение адресов чтения и записи.

Данные операции производятся с адресом чтения и записи отдельно. Порядок следования этих операций можно пронаблюдать на Рисунке 2.3.

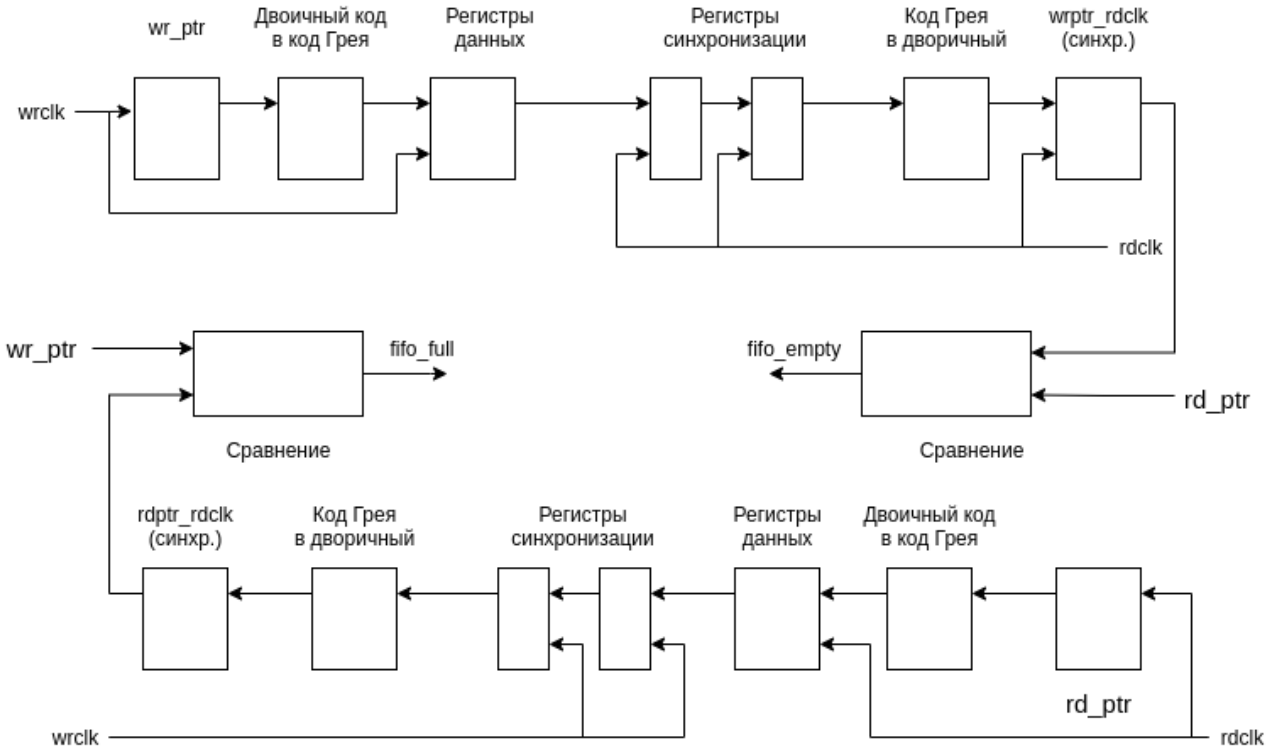

Рисунок 2.3 — Обработка адресов чтения и записи

2.5 Функциональная схема

Предлагается для построения устройства ресинхронизации данных, работающего в режиме очереди с независимыми тактовыми сигналами, использовать данную функциональную схему (см. Рисунок 2.4).

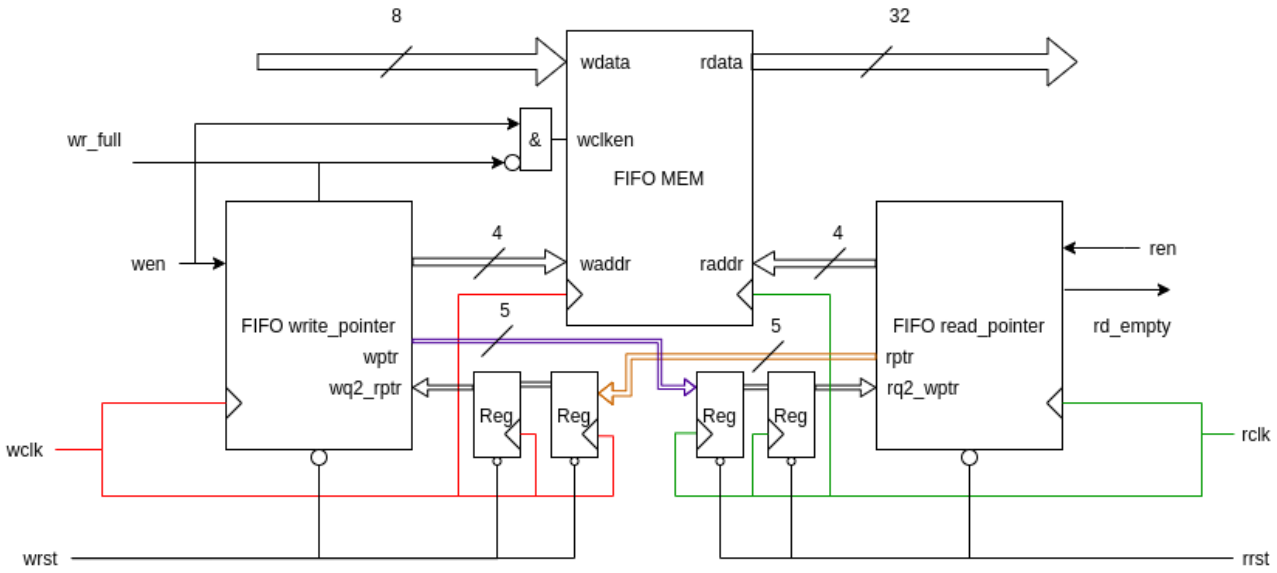

Рисунок 2.4 — Функциональная схема устройства

Приведем описание входов, выходов и служебных сигналов устройства (см. Таблицу 2.2).

Также укажем используемые глобальные и локальные параметры, используемые при реализации устройства на языке описания аппаратуры Verilog (см. Таблицу 2.1). С помощью данных параметров возможно произвести масштабирование устройства — увеличить разрядность ячейки запоминающего устройства и их количество.

Таблица 2.1 — Параметры

Название	Назначение	Значение
asize	разрядность адреса данных	4
dsize	разрядность данных в битах	8

Аппаратный блок по ресинхронизации данных разрабатывается для использования в совокупности с более сложными элементами и выполняет роль служебного блока хранения и передачи данных.

И

Таблица 2.2 — Сигналы входы и выходы устройства

Название	**Назначение**	**Разрядность**
mem	ячейки памяти очереди	[8:0] [0:15]
rbin	текущий адрес чтения в бинарном коде	[4:0]
rbinnext	адрес чтения в бинарном коде в следующем такте	[4:0]
rclk	тактирование на чтение данных	1
rdata	данные на чтение	[31:0]
rden	разрешение на чтение	1
rdredy	готовность выдать данные	1
rempty	сигнал пустоты очереди (прочтено все, что записано)	1
rempty_next	пустота очереди на следующем такте	1
rgray	текущий адрес чтения в коде Грея	[4:0]
rgraynext	адрес чтения в коде Грея в следующем такте	[4:0]
rq1_wgray	адрес чтения в первом синхро-триггере	[4:0]
rq2_wgray	адрес чтения во втором синхро-триггере	[4:0]
rrstn	сброс указателя чтения	1
wbin	текущий адрес записи в бинарном коде	1
wbinnext	адрес записи в бинарном коде в следующем такте	[4:0]
wclk	Тактирование на запись данных	1
wdata	данные на запись	[7:0]
wfull	сигнал заполнения очереди	1
wfull_next	заполненность очереди на следующем такте	1
wgray	текущий адрес записи в коде Грея	[4:0]
wgraynext	адрес записи в коде Грея в следующем такте	[4:0]
wq1_rgray	адрес записи в первом синхро-триггере	[4:0]
wq2_rgray	адрес записи во втором синхро-триггере	[4:0]
wren	разрешение на запись	1
wrstn	сброс указателя записи	1

3 РЕАЛИЗАЦИЯ ЦИФРОВОГО УСТРОЙСТВА

В данном разделе рассматривается процесс реализации устройства ресинхронизации данных и сопутствующих устройств средствами ISIM в САПР ISE Design Suite на языке описания аппаратуры Verilog.

3.1 Реализация модуля устройства ресинхронизации

Определим входные и выходные сигналы для модуля, реализующего устройство ресинхронизации данных (см. Таблицу 3.1).

Таблица 3.1 — Параметры файла afifo.v

Название	**Тип**	**Разрядность**
rden	input wire	1
rrstn	input wire	1
wclk	input wire	1
wren	input wire	1
wrstn	input wire	1
rclk	input wire	1
wdata	input wire	[dw-1:0]
wfull	output reg	1
rempty	output reg	1
rdready	output wire	1
rdata;	output wire	[(dw*4)-1:0]
wraddr	output wire	[aw-1:0]
rdaddr;	output wire	[aw-1:0]

Приведем содержание файла afifo.v, в котором содержатся описания блоков чтения, записи, тактирования и адресации устройства ресинхронизации данных в Приложении А на Рисунке А.1.

3.2 Реализация модуля устройства ресинхронизации

Для тестирования работы устройства опишем модуль, реализующий параметризованный делитель частоты. Приведем выходные и входные параметры данного модуля (см. Таблицу 3.2).

Таблица 3.2 — Параметры файла freq_div.v

Название	Тип	Разрядность
rst	input wire	1
clk	input wire	1
co	output reg	1

Исходный код данного модуля приведен в Приложении А на Рисунке А.2.

3.3 Тестирование модулей

После реализации необходимых модулей, произведем тестирование корректности работы устройств, описав тестовый файл afifo_test.v. Исходный код приведен в Приложении А на Рисунке А.3.

В качестве тестовых данных используем восьмиразрядные числа в шестнадцатеричном формате. Загрузку данных для удобства будем производить из файла data.txt. Чтение производится по 8 бит, тактирование по сигналу wclk. Запись производится в файл data_out.txt по 32 бита, при этом становится наглядным то, что устройство позволяет производить ресинхронизацию данных, выполняя роль шины данных между разными клоковыми доменами. Содержание тестовых файлов приведено в Приложении А на Рисунках А.4 и А.5.

Произведем симуляцию реализованных модулей средствами ISim в САПР ISE Design Suite. Результаты симуляции приведены в Приложении Б.

При включении реализуемого устройства в состав более сложных систем, предварительно необходимо ознакомиться с Руководством пользователя,

приведенным в Приложении В, в котором описаны особенности реализации и использования данного аппаратного блока.

ЗАКЛЮЧЕНИЕ

Результатом работы над данной курсовой работой является реализованное цифровое устройство ресинхронизации данных представляющее собой асинхронную очередь данных, тактируемое двумя независимыми синхросигналами, имеющее 8 разрядный вход и 32 разрядный выход.

В ходе данной работы были проведены мероприятия по разработке функциональной и структурной схемы цифрового устройства и устройств, необходимых для его работы; созданию модуля, реализующего устройство ресинхронизации данных на языке описания аппаратуры Verilog; включению данного устройства в состав сложных устройств и проведение его отладки и тестирования.

Цели и задачи по реализации устройства для ресинхронизации данных выполнены.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Приложение А Листинг

Листинг А.1 — Описание устройства ресинхронизации данных

```
1  module afifo(wclk, wrstn, wren, wdata, wfull,rclk, rrstn, rden, rdata,
       rempty, rdready,wraddr,rdaddr);
2
3  parameter  dsize = 8,
4             asize = 4;
5  localparam dw = dsize,
6             aw = asize;
7
8  input      wire  wclk, wrstn, wren;
9  input      wire  [dw-1:0] wdata;
10 input      wire  rclk, rrstn, rden;
11
12 output     reg   wfull;
13 output     reg   rempty;
14 output     wire  rdready;
15 output     wire  [(dw*4)-1:0] rdata;
16 output     wire  [aw-1:0] wraddr,rdaddr;
17
18             wire [aw-1:0] waddrmem, raddrmem;
19             wire [aw-1:0] waddrmem_mod4;
20             wire          wfull_next, rempty_next;
21             reg  [aw:0]   wgray, wbin, wq2_rgray, wq1_rgray,rgray, rbin,
                  rq2_wgray, rq1_wgray;
22
23
24             wire [aw:0] wgraynext, wbinnext;
25             wire [aw:0] rgraynext, rbinnext;
26
27             reg [dw-1:0] mem [0:((1<<aw)-1)];
28
29 //sync-w-rgray part
30 initial { wq2_rgray,  wq1_rgray } = 0;
31 always @(posedge wclk or negedge wrstn)
32  if (!wrstn)
33    { wq2_rgray, wq1_rgray } <= 0;
34 else { wq2_rgray, wq1_rgray } <= { wq1_rgray, rgray };
35
36 assign  wbinnext  = wbin + { {(aw){1'b0}}, ((wren) && (!wfull)) };
37 assign  wgraynext = (wbinnext >> 1) ^ wbinnext;
38 assign  waddrmem = wbin[aw-1:0];
39
40
```

```
41 assign wraddr=waddrmem;
42 assign rdaddr=raddrmem;
43
44 // wadress part
45 initial { wbin, wgray } = 0;
46 always @(posedge wclk or negedge wrstn)
47 if (!wrstn)
48     { wbin, wgray } <= 0;
49 else
50 { wbin, wgray } <= { wbinnext, wgraynext };
51
52 // comparison of low-order bits
53 assign  wfull_next = (wgraynext == { ~wq2_rgray[aw:aw-1],wq2_rgray[aw-2:0]
       });
54
55
56 initial wfull = 0;
57 always @(posedge wclk or negedge wrstn)
58  if (!wrstn)
59     wfull <= 1'b0;
60     else
61     wfull <= wfull_next;
62
63 always @(posedge wclk)
64     if ((wren)&&(!wfull))
65         mem[waddrmem] <= wdata;
66
67 //sync-r-wgray part
68 initial { rq2_wgray,  rq1_wgray } = 0;
69 always @(posedge rclk or negedge rrstn)
70 if (!rrstn)
71  { rq2_wgray, rq1_wgray } <= 0;
72 else
73 { rq2_wgray, rq1_wgray } <= { rq1_wgray, wgray };
74
75 // increment read address
76 assign  rbinnext  = rbin + { {(aw-2){1'b0}}, ((rden)?3'b100:3'b000)};
77 assign  rgraynext = (rbinnext >> 1) ^ rbinnext;
78
79 initial { rbin, rgray } = 0;
80     always @(posedge rclk or negedge rrstn)
81 if (!rrstn)
82     { rbin, rgray } <= 0;
83 else
84     if (wbin <2'b11)
85         rbin <=1'b0;
```

```
86      else
87      { rbin, rgray } <= { rbinnext, rgraynext };
88
89  assign  raddrmem = rbin[aw-1:0];
90  assign  rempty_next = (rgraynext == rq2_wgray);
91
92
93  initial rempty = 1;
94  always @(posedge rclk or negedge rrstn)
95  if (!rrstn)
96      rempty <= 1'b1;
97  else
98      rempty <= rempty_next;
99
100 //every fourth address in memory
101 assign waddrmem_mod4=(waddrmem-1'b1) & {{(aw-3){1'b1}},{3'b100}};
102 assign rdready = (waddrmem>=3'b100||!rempty);
103 assign  rdata = rden&&rdready?{mem[raddrmem],mem[raddrmem+1'b1],
104                 mem[raddrmem+2'b10],mem[raddrmem+2'b11]}:{(dw*4){1'b1}};
105
106 endmodule
```

Листинг А.2 — Описание устройства делителя частоты

```
1  module freq_div(input       rst,
2                  input       clk,
3                  output reg  co);
4                         reg [16:0] counter;
5       parameter divisior = 17'd4;
6
7       initial begin
8       co=0;
9       counter=0;
10      end
11
12      always @(posedge clk or posedge rst) begin
13         if (rst)
14             begin
15                 counter <= 0;
16                 co <= 0;
17             end
18         else
19                 if (counter >= (divisior - 17'b1))
20                     begin
21                         counter <= 17'b0;
22                         co <= 1;
23                     end
```

```
                else
                begin
                    co <= 0;
                    counter <= counter + 17'b1;
                end
    end
endmodule
```

Листинг А.3 — Описание тестового файла

```
module afifo_test();
parameter  dsize = 8,
           asize = 4;
localparam dw = dsize,
           aw = asize;
   integer fp_r,fp_w;
   reg                wclk, wrstn, wren,clk_rst;
   reg [dw-1:0]       wdata;
   wire               wfull,rempty;
   reg                rclk, rrstn, rden;
   wire  [(dw*4)-1:0] rdata;
   wire               freq_clk;
   wire               rdready;
    wire [aw-1:0]     wraddr,rdaddr;

freq_div #(17'd4) fr_dv (
          .rst(clk_rst),
           .clk(wclk),
          .co(freq_clk)
                    );

afifo ff(
      .wclk(wclk),
      .wrstn(wrstn),
      .wren(wren),
      .wdata(wdata),
      .wfull(wfull),
      .rclk (freq_clk),
      .rrstn(rrstn),
      .rden(rden),
      .rdata(rdata),
      .rdready(rdready),
      .rempty(rempty),
      .wraddr(wraddr),
      .rdaddr(rdaddr)
      );

```

```
38
39 always
40 begin
41    #5 wclk = ~wclk;
42 end
43 initial
44 begin
45   wdata=2'h01;
46   fp_r = $fopen ("data.txt", "r");
47   fp_w = $fopen ("data_out.txt","a");
48   clk_rst=0;
49   wclk=0;
50   wrstn=1;
51   rrstn=1;
52   wren=1;
53   rden=0;
54
55 #195;
56 rden=1;
57
58 #155;
59 wrstn=0;
60 #10;
61 wrstn=1;
62 end
63
64 always@(posedge wclk)
65 begin
66     $fscanf  (fp_r, "%h\n", wdata); // read one line
67 end
68
69 always@(posedge freq_clk)
70 if(rden)
71     $fdisplay (fp_w, "%h", rdata); // write to file
72
73 endmodule
```

Листинг А.4 — Входные данные

```
1 0a
2 0b
3 0c
4 0d
5 0e
6 0f
7 1a
8 1b
```

```
1c
1d
1e
1f
2a
2b
2c
2d
2e
2f
2f
```

Листинг А.5 — Выходные данные

```
010a0b0c
0d0e0f1a
1b1c1d1e
1f2a2b2c
2d2e2f2f
```

Приложение Б Результаты симуляции

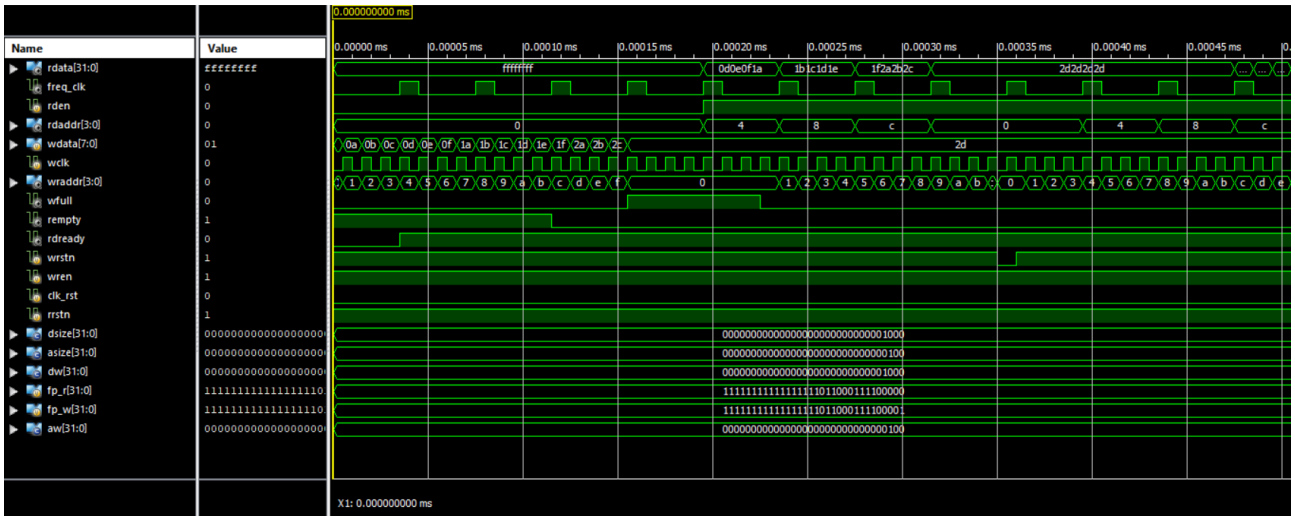

Рисунок Б.1 — Результат симуляции. Часть 1

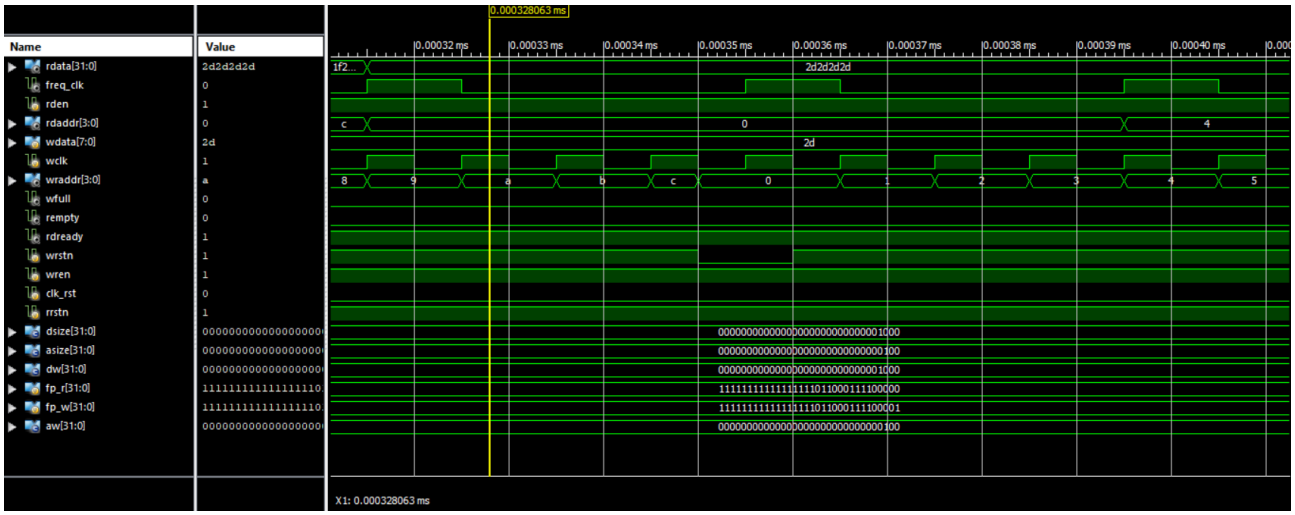

Рисунок Б.2 — Результат симуляции. Часть 2

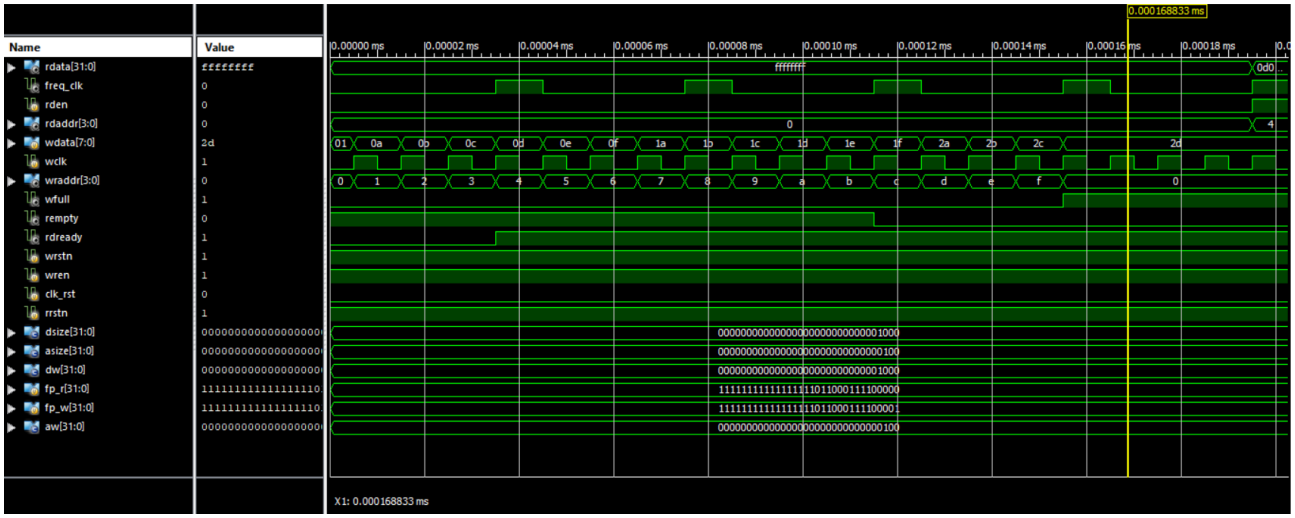

Рисунок Б.3 — Результат симуляции. Часть 3

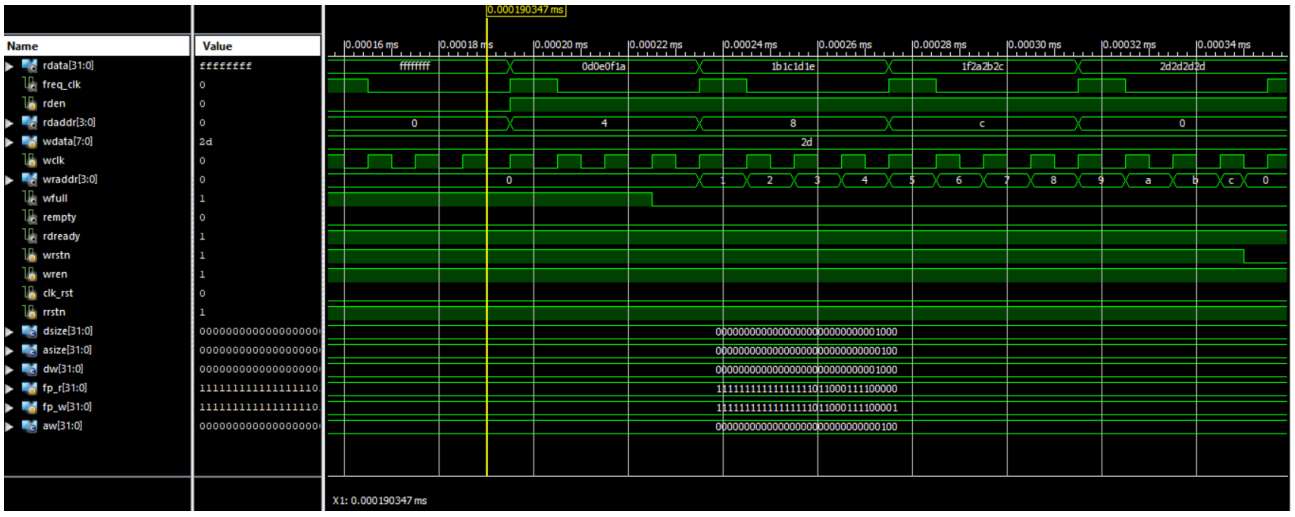

Рисунок Б.4 — Результат симуляции. Часть 4

Приложение В Руководство пользователя

В данном приложении приведены правила использования цифрового устройства по ресинхронизации данных.

Чтобы использовать аппаратный блок, спроектированный в ходе данной курсовой работы, в своих проектах, необходимо обратиться к списку выходов и входов, приведенному в разделе 2 настоящей курсовой работы и подключить модуль соответствующим образом.

Следует помнить, что данные на выход подаются по мере образования во входном буфере не менее четырех восьмиразрядных слов. Тактовые сигналы входного и выходного каналов независимы, однако тактовая частота для выхода устройства должна составлять как минимум четверть от тактовой частоты входного интерфейса устройства.

Восьмиразрядные числа, переданные на вход, образуют тридцатидвухразрядную выходную последовательность в прямом порядке следования битов (big endian).

Следует обратить внимание, что устройство осуществляет контроль записи и чтения — сигнализирует о пустоте или заполненности внутренней памяти, но не блокирует запись или чтение самостоятельно. Управлять разрешением на запись и чтение должны внешние узлы. Схема лишь не допускает обращения к одному участку памяти для записи и чтения единовременно.

Устройство не поддерживает сброс внутренней памяти, так как обычно память в устройствах, реализуемых на ПЛИС представлена в виде блочной оперативной памяти и выделенной оперативной памяти. Эти ресурсы обычно не поддерживают сброс. Сброс возможно реализовать в регистрах.