Федеральное государственное автономное образовательное учреждение высшего образования
«Московский физико-технический институт (национальный исследовательский университет)»
Физтех-школа радиотехники и компьютерных технологий
Кафедра электронных вычислительных машин

**Направление подготовки**: 03.04.01 Прикладные математика и физика (бакалавриат)
**Направленность (профиль) подготовки**: Радиотехника и компьютерные технологии

# ПРИМЕНЕНИЕ ВРЕМЕННЫХ МЕХАНИЗМОВ ВЗАИМОДЕЙСТВИЯ В АКТИВНОЙ ИЗМЕРИТЕЛЬНОЙ ПОДСИСТЕМЕ ЦИФРОВОЙ СИСТЕМЫ УПРАВЛЕНИЯ

(магистерская диссертация)

**Студент:**
Покровский Андрей Дмитриевич

_____
*(подпись студента)*

**Научный руководитель:**
Преображенский Николай Борисович,
Проф.

_____
*(подпись научного руководителя)*

**Консультант:**
Холопов Юрий Алексеевич,
ведущий инженер

_____
*(подпись консультанта)*

Москва 2022

# АННОТАЦИЯ

Рассматривается вариант построения централизованной цифровой информационной среды, оптимизированной для исполнения задач управления техническими объектами. Разработаны активные синхронные периферийные подсистемы цифровых систем управления с применением подхода согласованного во времени взаимодействия всех компонент. Показаны преимущества данного метода такие как возможность разнесения во времени конфликтующих процессов, одновременная фиксация параметров состояния и фазированная отработка параметров управления.

Подробно рассмотрена технология построения коммуникационного модуля - концентратора для соединения с датчиками, использующими широко распространенный в промышленности пакетный UART-интерфейс. Показана возможность встраивания модуля концентратора в систему на кристалле, реализованную в ПЛИС с использованием стандартной технологии Platform Designer из пакета Quartus Prime Lite Edition.

СОДЕРЖАНИЕ

# ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

СУ – система управления

ЦСУ – цифровая система управления

УСО – устройство сопряжения с объектом

ЭВМ – электронная вычислительная машина

ПЛИС – программируемая логическая интегральная схема

ЕПС – единый пакет состояния

АЦП – аналого-цифровой преобразователь

СЕВ – служба единого времени

ШИМ – широтно-импульсная модуляция

ЕПУ – единый пакет управления

# ВВЕДЕНИЕ

Для управления техническими объектами, широко используются разнообразные цифровые системы управления (ЦСУ). Формальной задачей систем управления (СУ) является преобразование набора параметров состояния объекта в набор параметров управления, необходимый для выполнения объектом его миссии. В аналоговых СУ процесс преобразования происходит постоянно и непрерывно, тогда как в ЦСУ преобразования параметров является одним из этапов следующей последовательности действий: измерение параметров состояния объекта управления и среды, в которой он находится; вычисление параметров управления на основе полученных параметров состояния и отработка параметров управления. Каждое из действий цепочки занимает определенное время, а вся цепочка повторяется периодически. Значения параметров состояния считаются неизменными до нового измерения в следующем цикле так же, как и параметры управления, после обновления сохраняются до нового цикла, поэтому можно назвать процесс преобразования параметров в ЦСУ дискретно-непрерывным.

Сгруппировав составные части ЦСУ по функциональному признаку, можно отнести их к одной из трёх подсистем: измерительной, расчетной и исполнительной (рис.1). Основным элементом расчетной подсистемы является вычислитель, а измерительной и исполнительной – устройства сопряжения с объектом (УСО): УСО-датчики и УСО-актуаторы соответственно. Все составные части ЦСУ объединены в единую систему цифровыми интерфейсами.

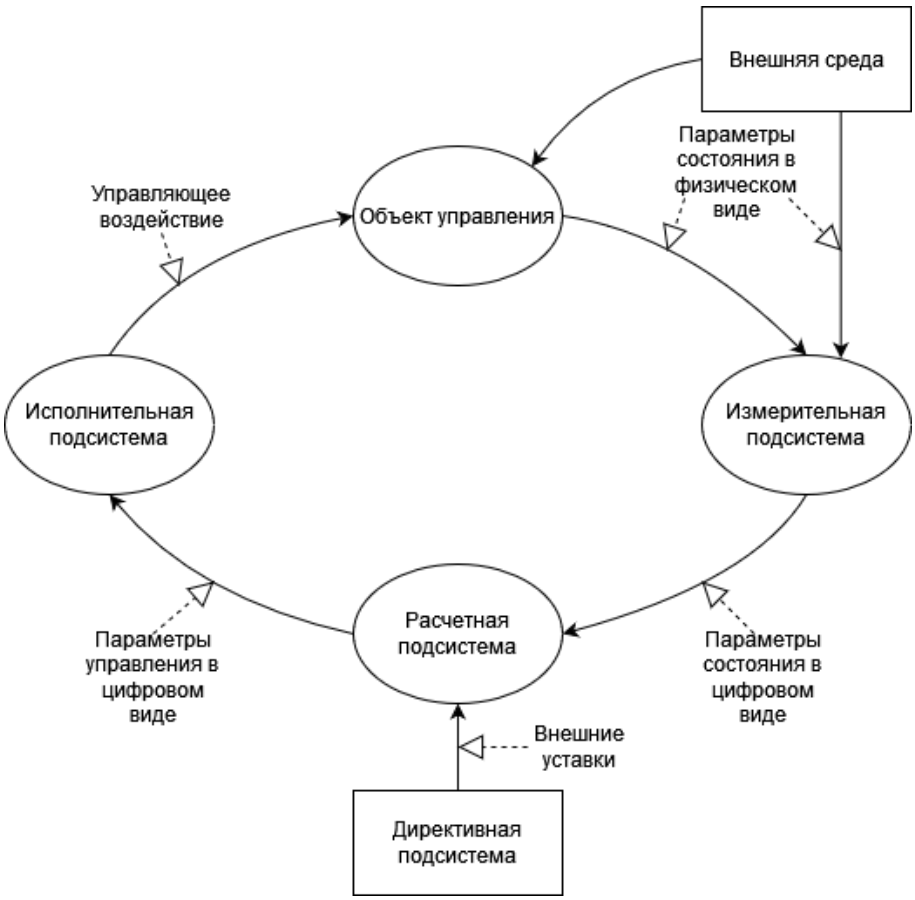



*Рисунок 1. Схема ЦСУ.*

В ЦСУ, построенных на универсальных вычислителях, основная часть измерительной и исполнительной подсистем реализована программно, за исключением самих датчиков и актуаторов. В связи с этим взаимодействие с периферией осуществляется посредством прерываний. Это приводит к тому, что выполнение программы управления объектом каждый раз останавливается, когда приходит очередное прерывание. В связи с этим, стоит отметить, что процесс выполнения алгоритма управления не является близким к его аналитическому описанию.

В существенной мере свойства ЦСУ определяются архитектурой ее вычислителя. Широко распространенные вычислители, которые сейчас используются для решения всех типовых задач, в том числе и задач управления, построены по архитектуре универсальных ЭВМ. Целью этой архитектуры является обеспечение максимальной загрузки самого дорогого узла в вычислительной машине - процессора. Поэтому стратегия управления

процессом вычислений процессора ориентирована на создание для него потока задач, которые, как правило, носят случайный характер.

Как отмечалось выше, в системах управления информационные процессы носят не случайный, а циклический характер. Это значит, что состав подзадач цикла управления постоянен и они повторяются периодически во времени. Период называется циклом регулирования, а задачи – списком функций цикла регулирования. Также фиксирован список операций ввода/вывода в каждом цикле регулирования, так как состав датчиков и исполнительных устройств неизменен, и они все опрашиваются каждый цикл регулирования. Таким образом, вычислители с архитектурой универсальных ЭВМ (далее – универсальные вычислители) могут быть оптимизированы для исполнения задач управления, с учётом статичных характеристик информационных процессов в ЦСУ и временных требований, о которых будет сказано ниже.

Универсальные вычислители, активно используют программное управление периферией, что является избыточно гибким для взаимодействия с работающими в циклическом режиме внешними устройствами ЦСУ. Также, в универсальных вычислителях невозможна программная реализация точных временных профилей управления УСО. Следовательно, в ЦСУ, операции ввода/вывода можно вывести из-под программного управления и передать некоторой аппаратной подсистеме, работающей по фиксированному циклическому алгоритму.

На заре развития вычислительной техники, стоимость аппаратуры многократно превышала стоимость разработки и хранения программ, поэтому было выгодно как можно больше операций управления периферией реализовывать программным способом. Подобное положение сохранилось в универсальных вычислителях до сих пор, хотя аппаратура стала намного дешевле и степень аппаратной поддержки операций ввода/вывода может быть существенно повышена.

Вынесение операций ввода/вывода в ЦСУ из-под программного управления приведет к тому, что программный алгоритм, который заложен в вычислителе, будет максимально соответствовать аналитическому описанию задачи управления объектом, так как возможна будет организация вычислений без прерываний.

Аппаратная реализация операций ввода/вывода работает в жёстком, фиксированном во времени режиме исполнения каждой операции и может использоваться как для формирования согласованного во времени управления измерениями разных параметров состояния, так и согласованной отработки параметров управления.

ОСНОВНОЕ СОДЕРЖАНИЕ

## 1. Влияние точного временного управления периферией на качество управления

Объясним необходимость согласованного управления датчиками в измерительной подсистеме ЦСУ. Одним из важнейших критериев будет являться возможность наиболее правильного и адекватного восприятия состояния объекта управления. Это можно достичь, если датчики будут работать в режиме «мгновенного снимка», т.е. фиксация параметров состояния будут происходить одновременно. Действительно, представим себе кадр из фильма (рис. 2), части которого были бы сделаны в разные моменты времени. Этот кадр будет разрывным, что исказит целостность всей картины.

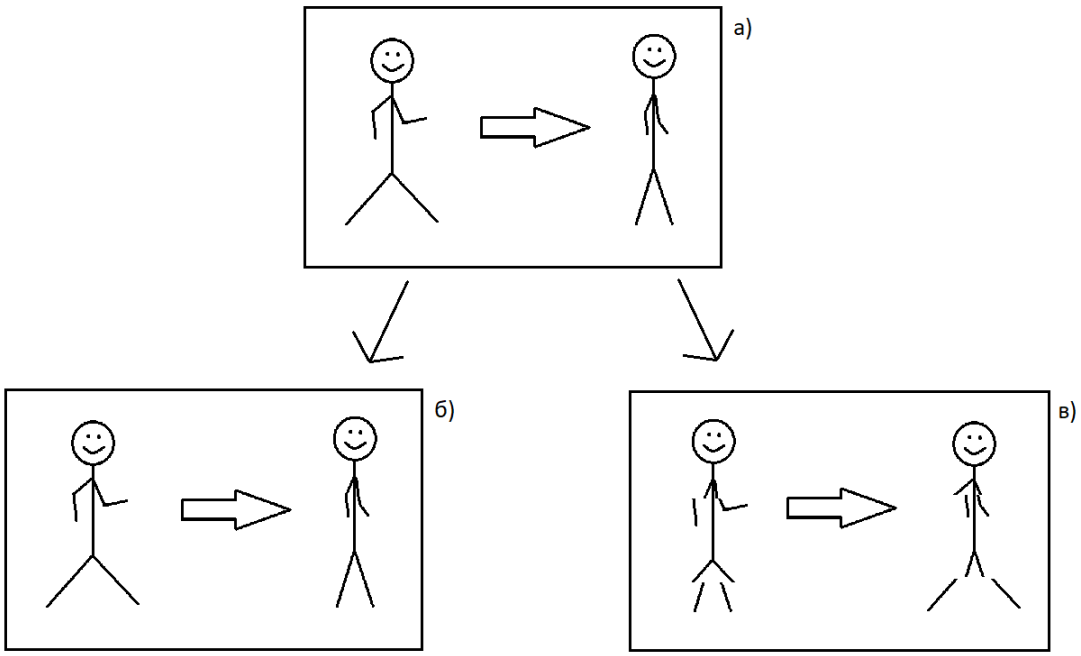



*Рисунок 2. Режим мгновенного снимка. а) - действие, которое происходит в действительности, б) - последовательность кадров при одновременной фиксации всех частей кадра, в) - последовательность кадров при фиксации частей кадра в разные моменты времени.*

В результате приходим к идее синхронной - одномоментной фиксации всех параметров состояния. Благодаря существованию цифровых интерфейсов

с жесткими временными диаграммами, становится возможной реализация синхронного управления даже пространственно распределённой периферией.

## 2. Проблемы применения универсальных ЭВМ в ЦСУ

Таким образом, потребовав одномоментной фиксации всех параметров, программная реализация операций ввода/вывода становится не только нерациональной, но и недопустимой из-за последовательного управления периферией в режиме прерываний. С их помощью реализуется основной способ обращения универсального ЭВМ к внешним устройствам. Обсудим это поподробнее. ЭВМ обрабатывает прерывания последовательно и каждое из них занимает некоторое время, которое, в общем случае, можно разбить на две части: первая - все, что касается передачи управления, и вторая - непосредственное взаимодействие с внешним устройством. Первую часть хотя и можно сократить с помощью дорогостоящих ОСРВ и высокопроизводительных машин, но ее длительность не детерминирована и в основном зависит от момента прихода запроса на прерывание и того, что в данный момент выполняет вычислитель. А вторая часть вовсе не зависит от производительности вычислителя и является характеристикой только внешнего устройства. В результате получаем, что из-за последовательной обработки прерываний фиксация параметров происходит с задержкой относительно друг друга (рис.3). Это можно было бы решить с помощью многоядерных вычислителей, но количество ядер современных стандартных ЭВМ много меньше количества внешних устройств в ЦСУ, которых может быть несколько десятков или даже сотен. К тому же необходимо ещё решать вопрос синхронизации работы ядер.

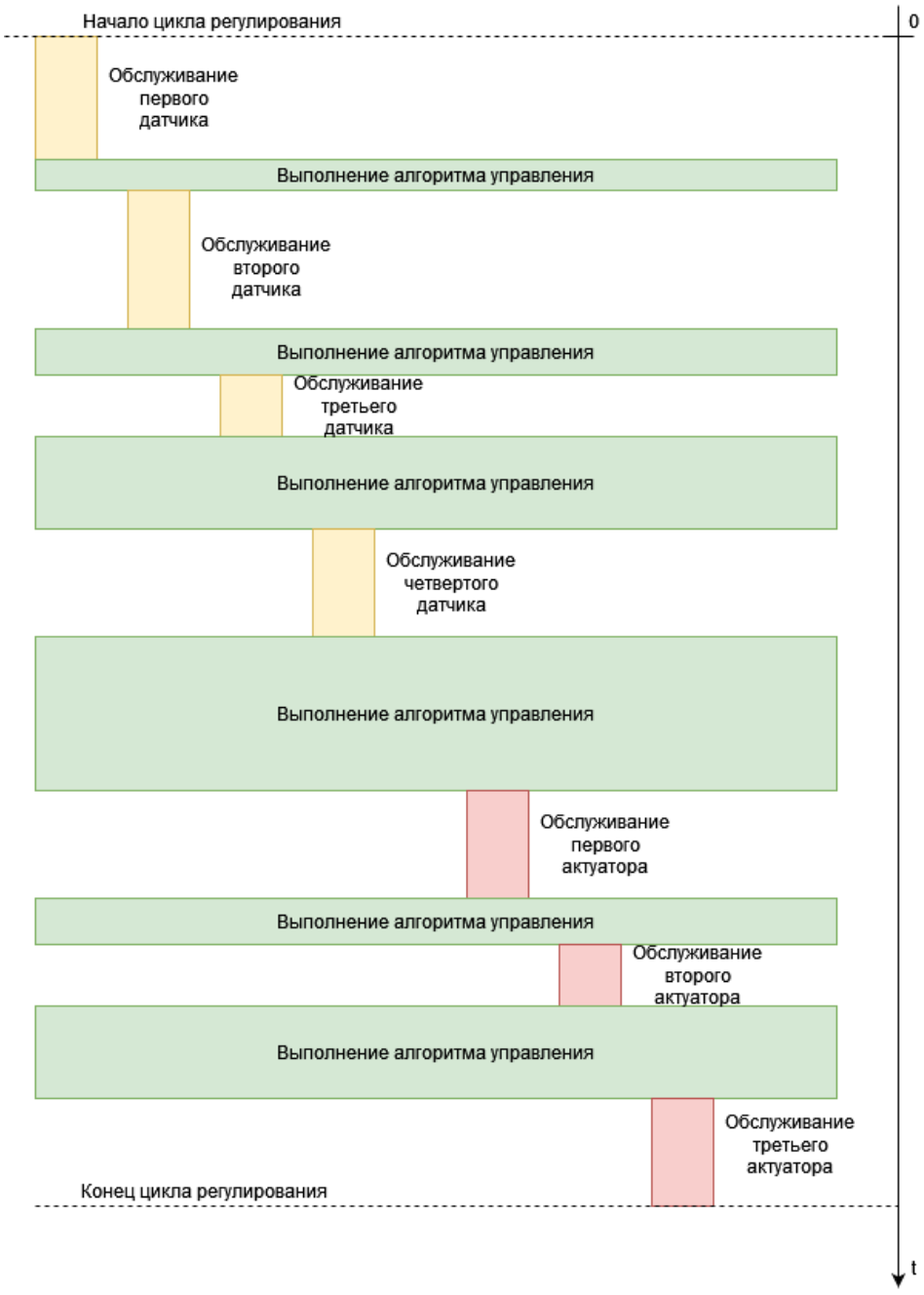



*Рисунок 3. Временная диаграмма программного управления периферией в ЦСУ.*

Еще одним аспектом, из-за которого не следует вычислители с архитектурой универсальных ЭВМ использовать в ЦСУ, является то, что они нацелены выполнять алгоритмы – последовательность действий, в которых нет такого параметра как время. А в ЦСУ все привязано ко времени: постоянная длительность цикла регулирования, когерентная фиксация параметров состояния. Время работы цифровых интерфейсов тоже

фиксировано, поэтому не рационально применять их в асинхронном режиме, который задают вычислители с архитектурой универсальных ЭВМ. Более приспособленными к выполнению задач управления являются вычислители, в которых учитываются и используются для технологической реализации элементы синхронности.

## 3. Анализ задачи построения активной синхронной измерительной подсистемы

Рассмотрим активную измерительную подсистему ЦСУ. Основная ее особенность заключается в том, что она работает автономно и вынесена из-под программного управления. Все ее операции запускаются по метке начала цикла, когда фиксируются параметры состояния в единый момент времени. Это возможно благодаря тому, что вся последовательность действий вплоть до записи параметров состояния в память вычислителя, статична, циклична и автономна относительно всех остальных внешних событий. Активная измерительная подсистема может являться как синхронной, где все процессы подчинены жесткому расписанию, так и полностью ей противоположной - асинхронной.

Целью данной работы является демонстрация возможности взаимодействия удаленных внешних устройств ЦСУ (в частности датчиков) с вычислителем в синхронном автономном режиме. Иными словами, нужно показать возможность создания ЦСУ с активной синхронной измерительной подсистемой. В этой работе будет показана попытка максимального использования временной синхронизации для: предотвращения конфликтов, создания режима когерентных измерений и фазированной выдачи уставок актуаторам.

Ключевым элементом должен быть построенный средствами ПЛИС модуль концентратора пакетов состояния, который будет выполнять функции: приема от датчиков параметров состояния, передаваемых по UART-каналам,

слияния этих данных в единый пакет состояния (ЕПС) и записи его в память расчетной подсистемы через DMA-канал, разовой, групповой операцией обмена.

## 4. Функции, исполняемые активной синхронной измерительной подсистемой

Для того, чтобы построить ЦСУ с активной синхронной измерительной подсистемой, необходимо реализовать следующие функции (рис. 4):

1. Циклически формировать временную метку начала очередного цикла управления.
2. Вначале каждого цикла управления фиксировать (запоминать) параметры состояния в единый момент времени.
3. Зафиксированные параметры преобразовать из аналогового вида в цифровой.
4. Переслать оцифрованные данные в концентратор параметров состояния.
5. Агрегировать параметры состояния в единый пакет состояния.
6. Передать ЕПС в память вычислителя разовой групповой операцией пересылки данных.

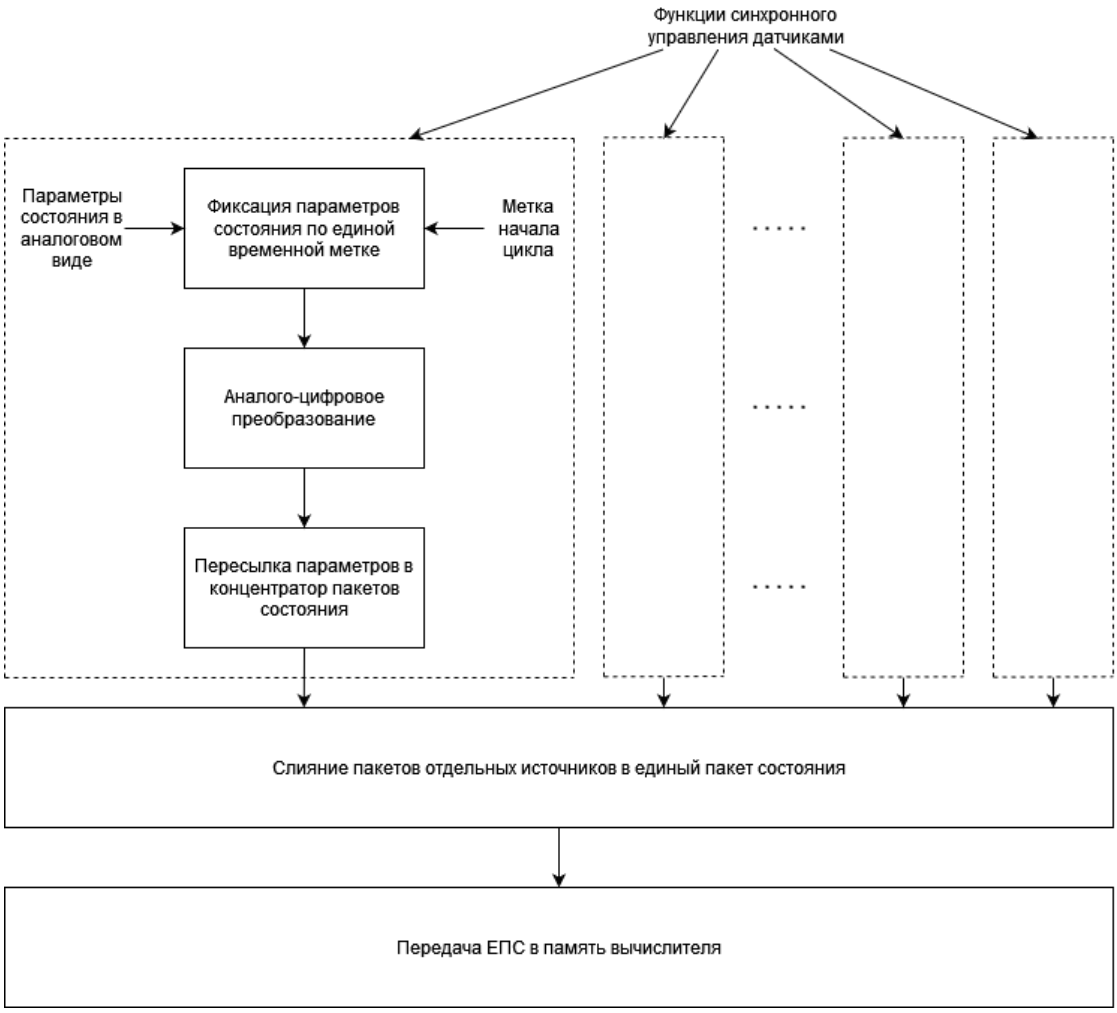



*Рисунок 4. Структурно-функциональная схема активной измерительной подсистемы*

## 5. Схема измерительной и расчетной подсистем

Теперь приведем одну из возможных реализаций набора функций, которые должна выполнять активная синхронная измерительная подсистема. Рассмотрим три основные части тракта ввода данных в вычислитель ЦСУ (см. рисунок 5). Первая часть, в которой реализуются измерение параметров, их фиксация и цифровизация, состоит из датчиков с АЦП, а также формирователя временных меток и службы единого времени (СЕВ), которые задают синхронный режим работы для всей системы. Второй частью является концентратор, в котором выполнена функция слияния пакетов от групп или отдельных датчиков в ЕПС.

В качестве канала передачи данных от АЦП в модуль концентратора был выбран UART-интерфейс, как самый распространенный цифровой пакетный

интерфейс в промышленности цифровых систем управления и как один из самых простых. Однако можно применять и более сложные цифровые интерфейсы, например Ethernet или I2C, которые, как и любые другие цифровые интерфейсы, имеют фиксированное время передачи пакетов, что является необходимым для организации синхронной работы составных частей ЦСУ. И третьей частью тракта ввода данных является память вычислителя, в которую записываются ЕПС из модуля концентратора пакетов состояния. Далее более подробно опишем весь состав и функционал измерительной подсистемы.

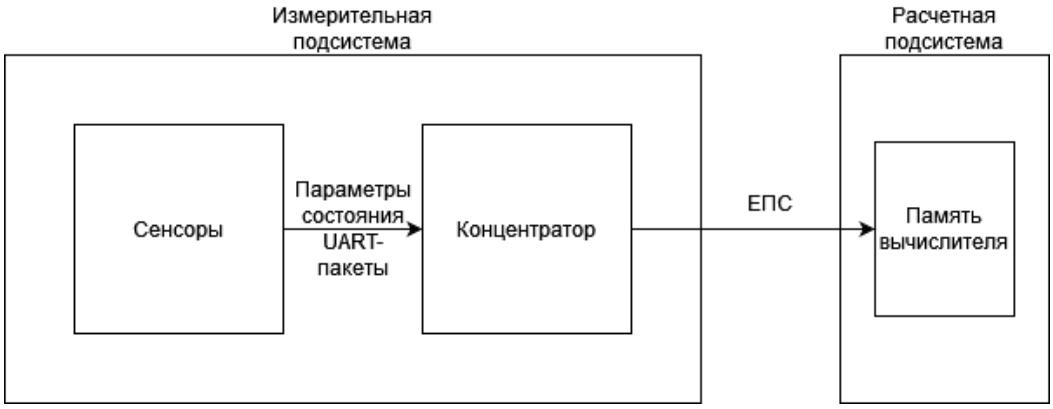



*Рисунок 5. Схема передачи параметров в расчетную подсистему*

### 5.1. Синхронное измерение параметров

В начале каждого цикла регулирования, через цифровые интерфейсы, к датчикам поступают обращения с командой о фиксации и цифровизации значений параметров в АЦП. Временные метки для запуска обменов формируются раз за цикл регулирования в формирователе временных меток таким образом, чтобы данные с разных датчиков были зафиксированы в АЦП одновременно. Иными словами, обращения к датчикам происходят с некоторым упреждением по времени относительно момента фиксации параметров, потому что необходимо учесть время на пересылку команды фиксации измеренных значений (рис.6). Так как времена обменов через цифровые интерфейсы и длительность цикла регулирования фиксированы, можно вычислить моменты, когда необходимо обратиться к тому или иному датчику. Формирователь временных меток сравнивает значение текущего

времени, определяемое СЕВ, с некоторыми константными значениями, чтобы в нужный момент запускать обмены (рис.7).

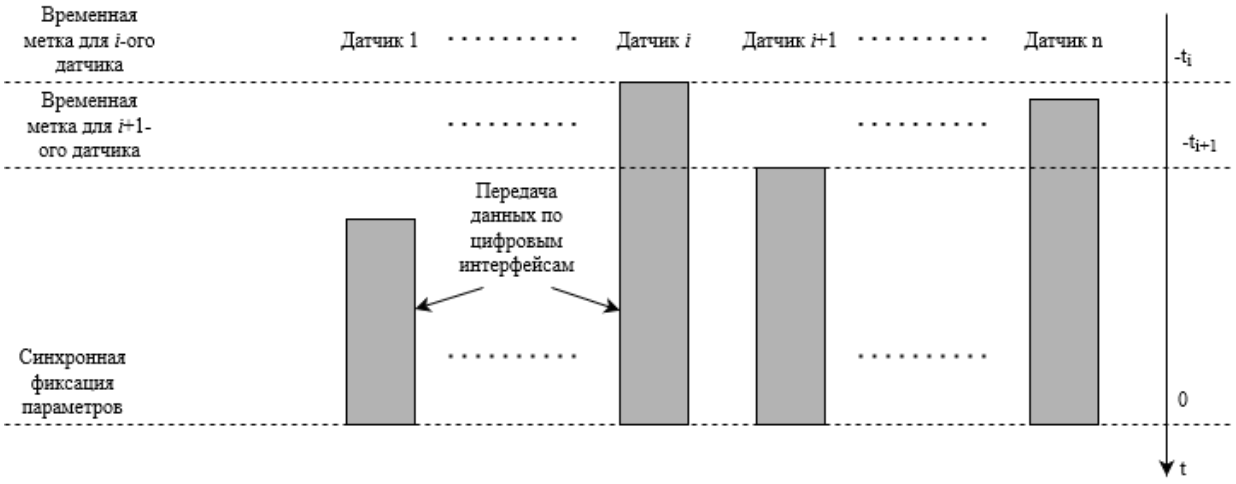



*Рисунок 6. Обращение к датчикам с упреждением по времени относительно фиксации параметров.*

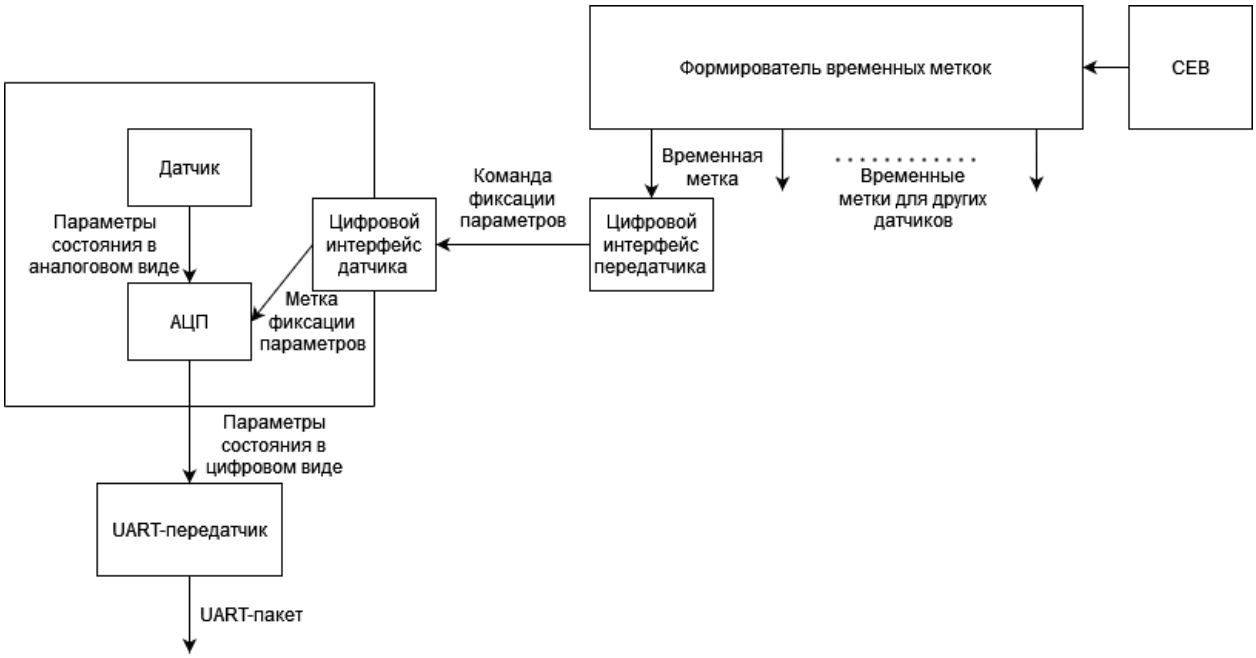



*Рисунок 7. Схема синхронной фиксации параметров.*

### 5.2. Слияние параметров в ЕПС

Как только все параметры состояния были зафиксированы в АЦП и преобразованы в коды, их необходимо переслать в модуль концентратора пакетов состояния через выбранный к использованию в проекте UART-канал. Следовательно, в концентраторе должны быть реализованы UART-приемники. Принятые UART-пакеты необходимо проверять на целостность и

собирать из них единый пакет состояния. Пока идет проверка на целостность принятых данных, их необходимо помещать в буфер для временного хранения. Так как несколько UART-приемников будут передавать данные в единый буфер – формирователь ЕПС, то требуется применять механизм многоканального доступа к памяти (рис. 8).

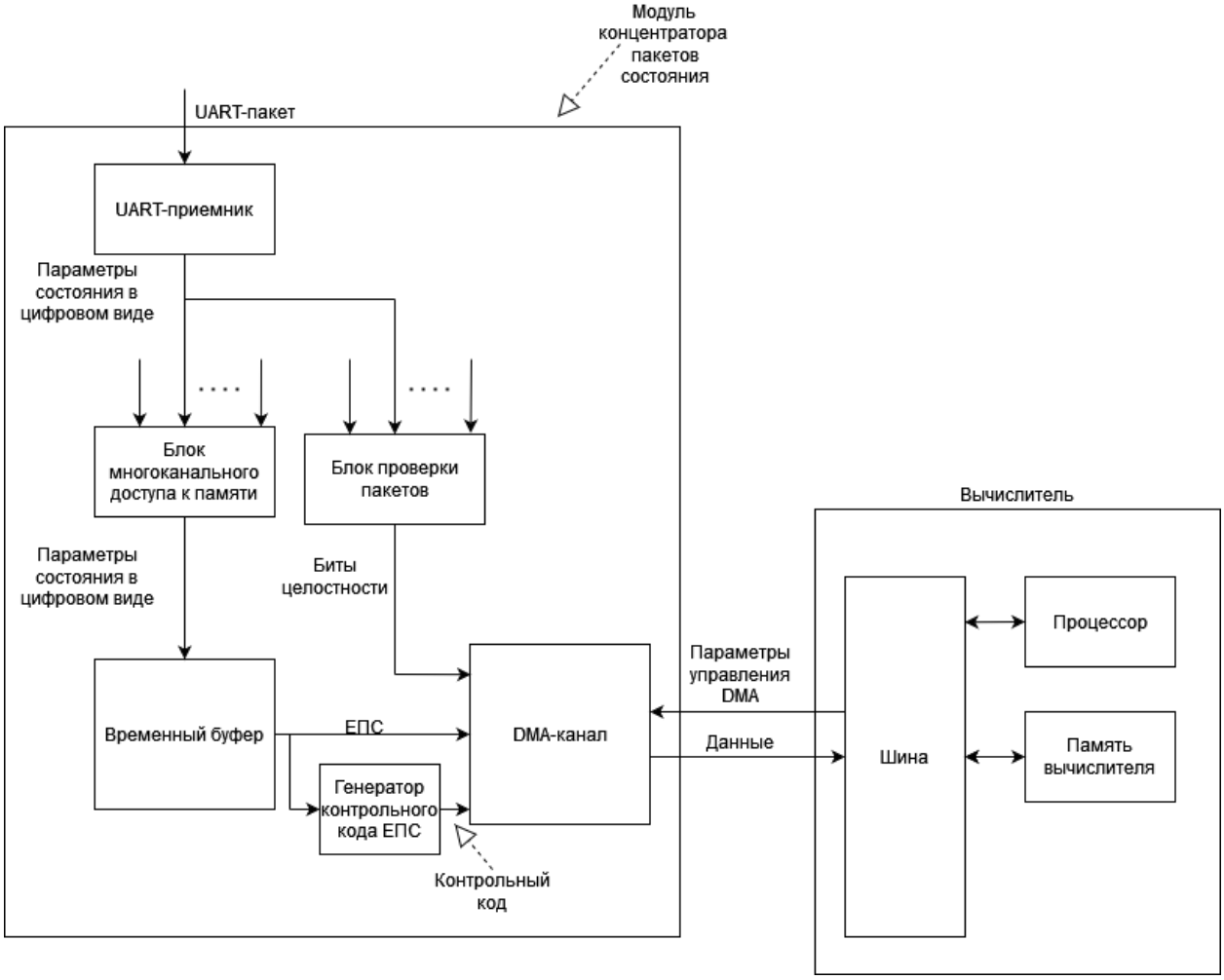



*Рисунок 8. Схема слияния параметров состояния в ЕПС и передачи его в память вычислителя.*

### 5.3. Запись ЕПС в память вычислителя

Завершающим этапом ввода данных в расчетную подсистему будет пересылка ЕПС и информации о том, пакеты от каких датчиков были получены с ошибкой (биты целостности), из единого буфера в память вычислителя через DMA-канал. Также в память вычислителя будет передана контрольная сумма ЕПС и битов целостности. Для случая, если будут ошибки

в каналах передачи, предусмотрено, что в памяти вычислителя будет две области, в которые будут записываться ЕПС поочередно. Если были ошибки, то в качестве параметров состояния будут взяты параметры из предыдущего цикла регулирования, когда параметры были достоверны.

### 5.4. Формирователь временных меток

На рисунке 6 показана логическая схема соединения формирователя временных меток и датчиков. Она отражает принцип их взаимодействия и тот факт, что появление меток фиксации параметров зависит только от времени и ни от никаких других событий, но неверна с точки зрения физического соединения. На этом рисунке датчики имеют якобы два интерфейса: с формирователем временных меток и UART-передатчиком.

Рассмотрим два варианта корректного соединения формирователя временных меток с датчиками. Их выбор зависит от размеров объекта управления. Если он достаточно мал, то есть его габариты не превышают длину шин интерфейсов, которые используют датчики, то формирователь временных меток будет выполнять функции коммутации с датчиками и преобразования полученных данных от датчиков для передачи их по UART-каналам. Таким образом, получим, что формирователь временных меток будет содержать в себе еще модули управления датчиками и формирования UART-пакетов (рис.9).

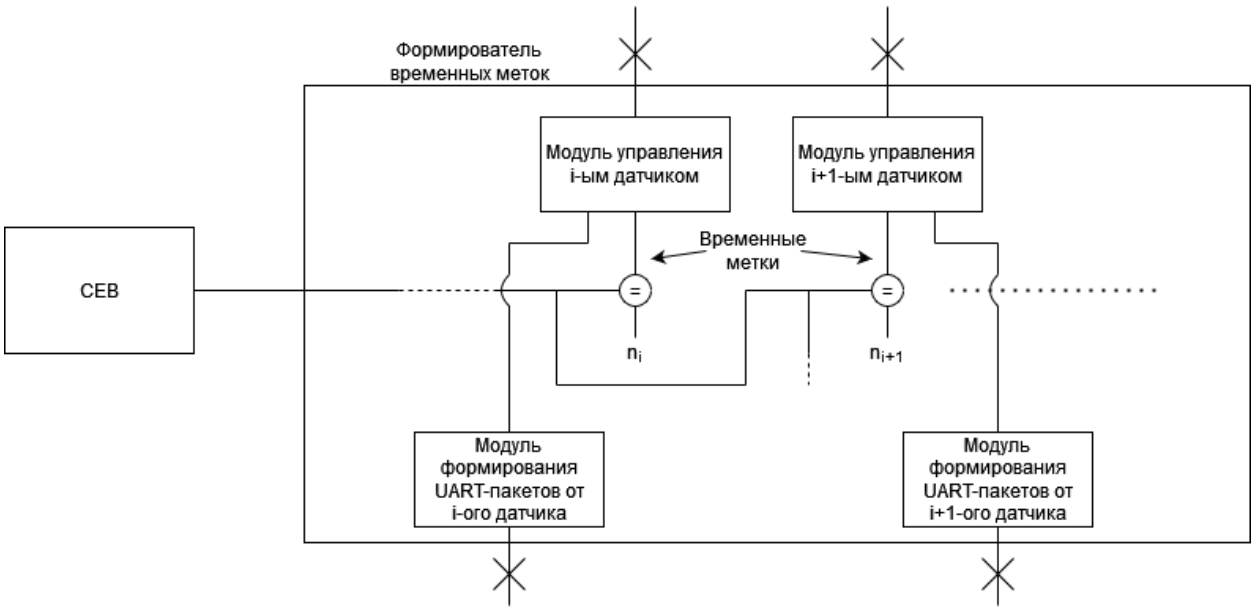



*Рисунок 9. Формирователь временных меток для малого объекта управления*

В случае, когда размеры объекта управления превышают максимальную длину шины интерфейса хотя бы одного датчика, формирователь временных меток следует внести в концентратор, а модули управления датчиками распределить по объекту управления, расположив их рядом с соответствующими датчиками (рис.10). В результате получим, что модули управления датчиками будут подключаться к концентратору как удалённые устройства, в нашем случае – через пакетный UART-интерфейс. В этом случае при формировании временных меток придется учитывать время на пересылку уже по двум интерфейсам и время преобразования формата пакетов. Данные от датчиков пойдут по тому же пути, только в обратном направлении. Стоит также отметить, что этот вариант подойдет и для управления малыми объектами, но потребуется больше расчетов, чтобы вычислить задержку.

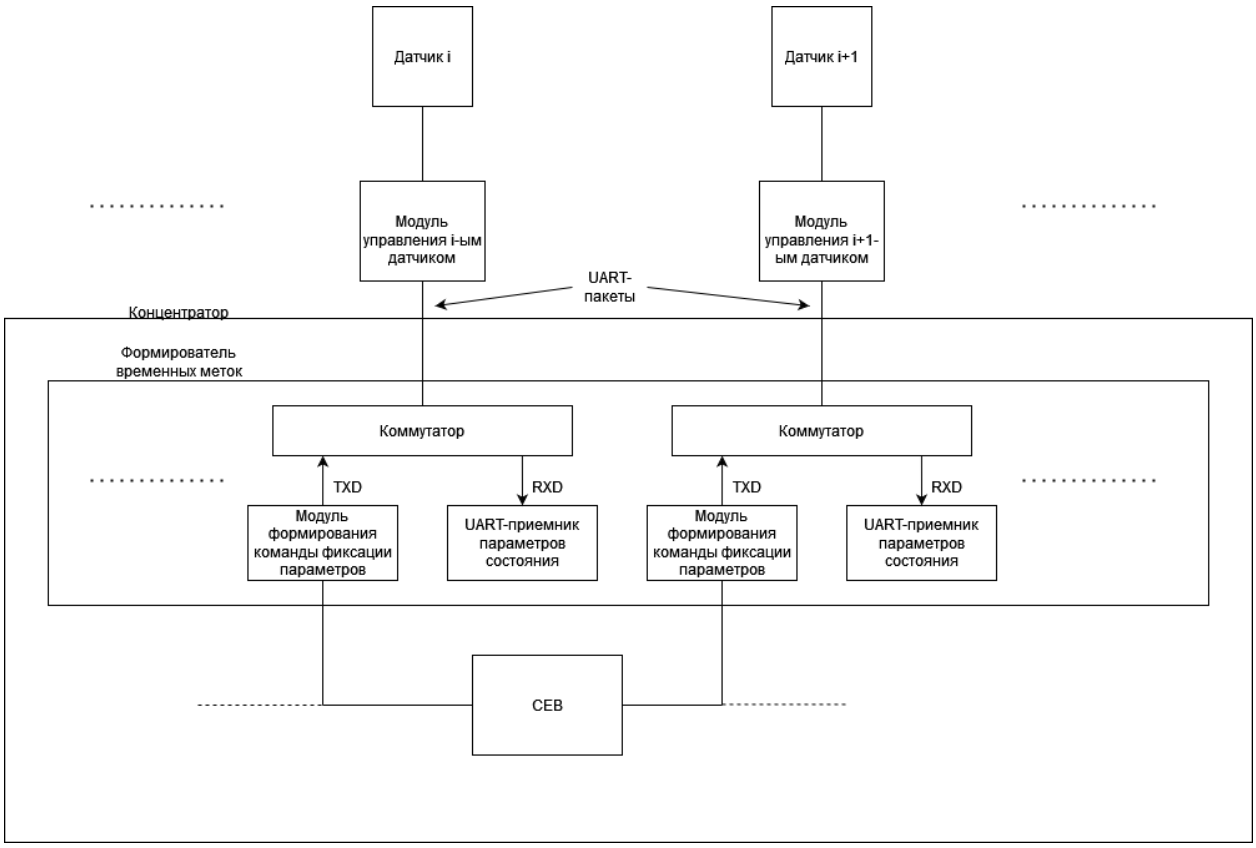



*Рисунок 10. Формирователь временных меток для крупных объектов управления.*

### 5.5. Вычисление моментов формирования временных меток

Из документации по интерфейсам можно найти время, за которое предаются те или иные команды. Это время также можно вычислить, зная, сколько предается битовых интервалов и скорость передачи данных. Однако, необходимо также учитывать задержки в каналах связи, чтобы точно определить моменты формирования временных меток. Для того, чтобы вычислить полное время передачи команд используем следующий подход. Помимо основного режима работы измерительной подсистемы, когда происходит измерение и передача параметров, введем еще один вспомогательный, для вычисления полного времени передачи команд. В этом режиме будем использовать технологический режим канала связи, в котором происходит немедленное отражение байта назад. Таким образом, отправив один пакет от формирователя временных меток, можно пересылать его к датчику и обратно несколько раз. Затем, замерив время на его пересылку,

разделим его на значение, равное количеству пересылок, получим усредненное время передачи команды в обе стороны, поэтому это время разделим еще пополам, считая, что задержки в обе стороны равны, и тогда найдем полное время на пересылку команды в одну сторону. В случае, если модуль взаимодействия с датчиком будет прерывать линию передачи данных, то можно просто вычесть время на изменение формата данных из времени на многократную пересылку из-за того, что изменение форматов для разных направлений может быть разным, а в конце его прибавить, потому что это время фиксировано и точно определяется при создании модуля.

Измерение времени на передачу команды фиксации необходимо запустить до начала работы основного режима измерительной подсистемы. Также, возможно, иногда потребуется выполнение вспомогательного режима во время работы ЦСУ из-за изменения условий окружающей среды, например температуры, что может привести к изменению времени задержки. Тогда требуется эти режимы разнести во времени, а о начале режима немедленного отражения байта сообщать датчикам через специальную команду, которую также выдаст формирователь временных меток. Эту команду можно формировать одновременно для всех датчиков без кого-либо упреждения, здесь точность не нужна. Измерение времени на передачу команды фиксации можно проводить, например, когда идет отработка уставок.

## 6. Реализация модуля концентратора

Последовательно опишем структуру и функционал узлов, образующих модуль концентратора пакетов состояния.

### 6.1. Функция приема данных

Как было сказано выше, задачу приема данных в блок концентратора выполняют UART-приемники. Для начала определим формат UART-посылок в данной работе: стартовый бит, 8 бит данных и стоповый бит.

Измеряемые параметры могут иметь значения, которым требуется разрядность больше, чем 8 бит, поэтому необходимо сформировать пакеты, состоящие из нескольких таких посылок. Как в случае любой пакетной передачи, для проверки целостности UART-пакетов необходимо еще передавать контрольный код. В данной работе он будет занимать 2 последних байта в каждом пакете.

Теперь представим интерфейс UART-приемников и их основные параметры (табл.1).

*Таблица 1 - Интерфейс UART-приемника*

| **Имя сигнала** | **Разрядность** | **Тип** | **Функция** |
|---|---|---|---|
| UART-channel | 1 | Вход | Данные с линии |
| unload | 1 | Выход | Сигнал, сообщающий о передаче принятых данных во внешние модули |
| DRx | 8 | Выход | Полученные данные |
| next | 1 | Вход | Сигнал, разрешающий передачу принятых данных во внешние модули |

Для того, чтобы достоверно принимать UART-посылки, нужно, что бы приемник был настроен на ту же скорость передачи данных, что и передатчик. Так как UART является интерфейсом без выделенного сигнала синхронизации, то приемник должен сам выполнять синхронизацию – распознавать начало UART-посылки с необходимой точностью. Для этого требуется разделить битовый интервал на 16 равных частей и завести счетчик, который будет их считать. Период счётчика равен длительности одного бита. Синхронизация счётчика длительности производится по переднему фронту стартового бита. Также требуется еще один счетчик для определения количества принятых бит, чтобы определить конец посылки. Таким образом, точность определения начала пакета не будет превышать 1/16 битового интервала, что соответствует стандартам де факто работы с UART-интерфейсом.

Данные будут приниматься в сдвиговый регистр в определённой фазе битового интервала, формируемой по фиксированному состоянию счётчика длительности. После приёма данных UART-посылки, они переписываются в выходной регистр, откуда считываются внешними модулями.

## 6.2. Функция агрегирования данных в ЕПС

Опишем, как происходит слияние данных из разных UART-пакетов в единый массив. Для реализации функции агрегирования данных в ЕПС необходимы временный буфер (реализуем его как двухпортовую RAM память, которая на схеме обозначена как IP_block_Mem), в который записываются принимаемые данные, и блок многоканального доступа к памяти, который будет регулировать процессы записи и чтения от нескольких запросчиков, которыми могут являться, например UART-приемники. Параллельно с записью во временный буфер должна производиться проверка целостности принятых данных. Эта проверка осуществляется путем анализа контрольной суммы. Данные, передаваемые в буфер, также должны поступать на вход блока проверки контрольной суммы (на схеме check_CRC_i), причем сама контрольная сумма не должна передаваться в буфер. После проверки должны взводиться триггеры (назовем их битами целостности), если данные были получены верно. Они нужны для того, чтобы при анализе параметров состояния было известно, какие из них достоверны.

### 6.2.1. Структура временного буфера

Поясним структуру временного буфера. Область памяти буфера требуется разделить на равные части, количество которых определяется следующим образом. Пусть есть некоторое количество датчиков, мысленно дополним это число до ближайшей большей степени двойки. Полученное число будет равно количеству частей, на которые делится память. Каждая часть памяти предназначена для хранения данных от одного датчика, но могут остаться части, которые не будут заполнены. Такое разбиение памяти удобно для обращения к ней. (Пример. Пусть всего 23 датчика, тогда дополняем это

число до ближайшей большей степени двойки, это будет 32, т.е. всю память разделим на 32 части).

Рассмотрим интерфейс двухпортовой RAM. Входными сигналами являются: адреса записи и чтения, данные записи (разрядностью 8 бит в соответствии размером UART посылки), разрешения на запись и чтения и тактовый сигнал. Выход только один – данные чтения, тоже 8 бит.

Определим, как будет формироваться адрес для обращения к временному буферу (рис.11). Пусть размер памяти равен $2^n$ байт и количество датчиков равно m. Адрес будет состоять из полей base и offset. В поле base будет записан базовый адрес, значение которого будет определяться номером запросчика. По этому адресу будет записываться первый байт данных из соответствующего канала. offset определяет смещение относительно базового адреса, и это смещение равно номеру байта, передаваемого в канале (нумерация байтов начинается с 0). Таким образом, получаем, что разрядность шины адреса равна n, адрес нулевого байта для каждого канала равен $k*2^{n-p}$, где p – показатель степени числа с основанием 2, которое получилось в результате дополнения числа датчиков до ближайшей большей степени двойки, а k – номер запросчика. Для базового адреса выделяется p старших разрядов, и n-p младших адресов - для смещения.

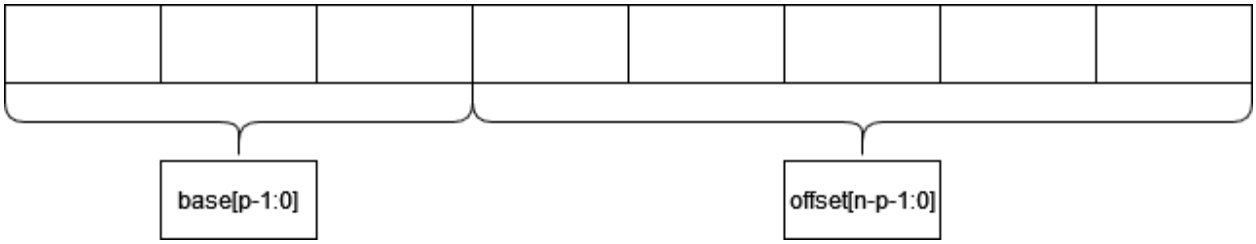



*Рисунок 11. Формат адреса временного буфера.*

### 6.2.2. Блок многоканального доступа к памяти

После того, как UART-приемником, принимается очередной байт его необходимо записать в коммутационный буфер. Для модуля двухпортовой RAM, UART-приемник выступает в качестве запросчика к ресурсам памяти. А так как этих запросчиков несколько то, следовательно требуется реализовать функцию многоканального доступа к памяти для этих абонентов. Так же запросчиком является канал DMA.

Опишем в виде таблицы интерфейсы блока многоканального доступа к памяти с памятью и запросчиками (табл.2).

*Таблица 2 - Интерфейсы модуля многоканального доступа к памяти*

| Имя сигнала | Интерфейс | Разрядность | Тип | Функция |
|---|---|---|---|---|
| unload[i] | С запросчиком | 1 | Вход | Сигнал, сообщающий о приеме данных от UART-приемника (i – номер запросчика) |
| DRx[i] | С запросчиком | 8 | Вход | 1 байт данных, принятые UART-приемником |
| op[i] | С запросчиком | 1 | Вход | Тип операции: запись или чтение |
| next[i] | С запросчиком | 1 | Выход | Пока он равен 0 (предыдущий запрос абонента еще не был обработан), сигналы и данные из UART-приемника не проходят в блок многоканального доступа к памяти |
| data_in | С памятью | 8 | Выход | Данные, записываемые в память |
| wr_en | С памятью | 1 | Выход | Разрешение на запись в память |
| wr_addr | С памятью | n | Выход | Адрес записи |
| rd_en | С памятью | 1 | Выход | Разрешение на чтение из памяти |
| rd_addr | С памятью | n | Выход | Адрес чтения |
| data_out | С памятью | 8 | Вход | Данные, считываемые из памяти |

На вход блока многоканального доступа к памяти поступают сигналы с выходов UART-приемников (на схеме в приложении «схема концентратора» это UART_Rx_i), среди них: unload[i] – импульсный сигнал, сообщающий о приеме данных от UART-приемника (i – номер запросчика), и DRx[i] – 1 байт данных, а также тип операции – op[i], если 1, то запись, иначе чтение. Сигналы unload[i] возводят соответствующие триггеры ASK[i], что означает, что есть очередные запросы от соответствующих абонентов, которые необходимо обслужить. Однако, если принимаются части контрольной суммы, то установка триггеров ASK[i] не производится. В блоке многоканального доступа к памяти также есть регистры данных размером в 1 байт, в которые по сигналу unload[i] записываются данные DRx[i]. Дополнительно укажем, что, если в UART-приемнике сформирован очередной байт данных, но при этом предыдущий байт от того же приемника еще не был записан в буфер, то новый так и останется в UART-приемнике, пока старый не будет записан. Для реализации этого введен сигнал next[i], который является входом UART-приемника. Пока он равен 0, сигналы и данные из UART-приемника не проходят в блок многоканального доступа к памяти. В качестве сигнала next[i] можно использовать отрицание сигнала ASK[i].

Так как данные поступают в произвольные моменты времени, могут возникать конфликты при записи, поэтому в блоке многопортового доступа к памяти должна быть реализована логика поочередного обращения запросчиков к памяти. Будем рассматривать логику фиксированных приоритетов абонентов. Если пришло несколько запросов на запись, иными словами, взведено несколько триггеров ASK[i], то обслуживается первым тот запросчик, чей номер меньше. Их номера фиксированы и определяются при проектировании как порядковые (1, 2, 3…). Такой подход оправдан, так как время между UART-посылками достаточно велико, чтобы все данные одной посылки были записаны во временный буфер до прихода следующей.

Для реализации этой системы необходимо сформировать два вектора SEL[] и PV[] длины m (количество датчиков) (рис.12). Каждому разряду этих векторов ставится в соответствие один абонент по принципу: самому младшему разряду – абонент с самым маленьким номером. Для поиска запросчкика с наибольшим приоритетом требуется вектор PV[], разряды в котором заполняются рекуррентно. Покажем, как реализована данная логика. Единицы в этих разрядах будут устанавливаться, если нет запросов от абонентов с меньшим номером, поэтому младший разряд PV[] всегда будет равен 1, так как нет запросчика с меньшим номером. Таким образом, разряд PV[], в котором стоит старшая единица, будет соответствовать выигравшему абоненту т.е., от которого пришел запрос и который обладает наибольшим приоритетом. Затем активным устанавливается только 1 бит вектора SEL[] – у выигравшего запросчика. Положение этой единицы и будет определять абонент, данные из которого будут записаны в память.

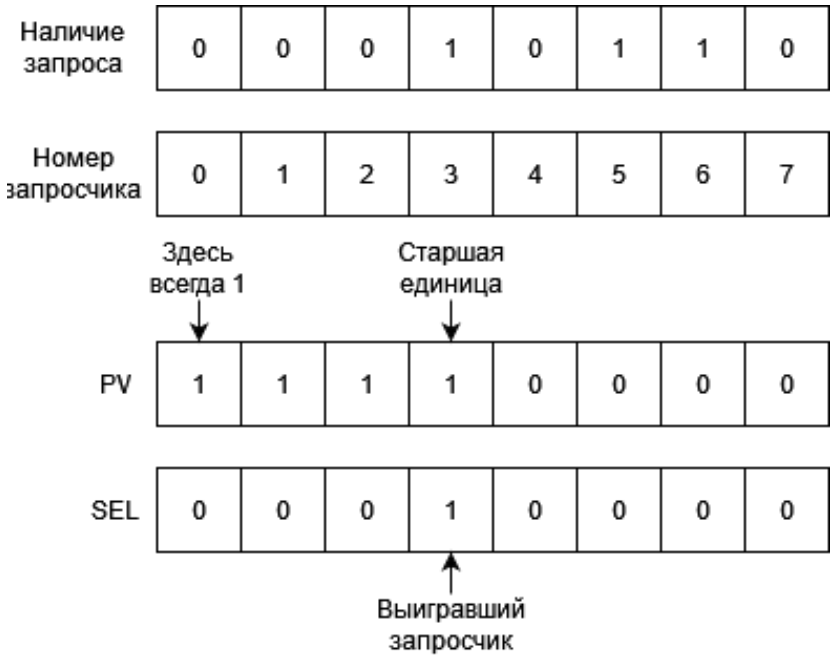



*Рисунок 12. Пример состояния векторов PV и SEL для случая с 8 абонентами.*

Опишем эту логику на языке verilog.

PV[0] = 1;//наименьший номер запросчика, поэтому этот разряд всегда равен 1.

PV[1] = PV[0] & !ASK[0];//если у запросчика с б*о*льшим приоритетом нет запроса, то следующий разряд PV[] также становится равным 1.

PV[i] = PV[i-1] & !ASK[i-1];// если у всех запросчиков с б*о*льшим приоритетом нет запросов, то следующий разряд PV[] после всех, от 0-ого до i-1-ого, также становится равным 1.

А теперь опишем вектор селектора:

SEL[i] = ASK[i] & PV[i];//это значит, что если есть запрос у i-ого запросчика (взведён триггер ASK[i]) и у всех запросчиков с меньшим номером нет запроса, то для обслуживания выбирается i-ый запросчик.

Теперь поясним, как с помощью вектора селектора выбрать нужные данные, адрес и тип операции. Рассмотрим пример с данными (для адреса и типа операции аналогично). Для этого произведем операцию И между всеми битами данных, хранящихся в i-ом регистре, и i-ым битом вектора селектора, и так сделаем для каждого регистра. Затем результаты объединим по ИЛИ. Так как в каждый момент времени единица в векторе селектора может находиться не больше чем в одном разряде, то данные, поступающие на вход ИЛИ, будут равны нулю, кроме того канала, номер которого соответствует положению единицы в векторе селектора. Таким образом, на выходе ИЛИ получим данные от выигравшего запросчика для записи в память.

Так как операции в двухпортовой памяти занимают один такт, то сигнал SEL[i] можно рассматривать как сигнал, выполняющий в данном месте функцию сброса триггера ASK[i], причем установка ASK[i] обладает большим приоритетом чем сброс. Данный подход позволяет достичь максимального быстродействия (отсутствие пауз между обращениями к памяти).

Теперь определим сигнал разрешения записи в память – wr_en. Когда пришел только первый запрос на запись, то разрешение записи определяется началом сигнала, который обозначает, что есть хотя бы один запрос. Если

запросов несколько, то запись будет разрешена, если есть единица в векторе SEL и в регистре op[i] записана единица.

Укажем, как формируется адреса записи и чтения. Они состоят из двух регистров, объединенных в один счетчик: номер абонента и номер смещения байта для этого абонента, причем номер абонента – это старшие разряды счетчика, а номер смещения – это младшие разряды.

### 6.2.3. Разнесение пакетов состояния во времени

Можно предложить еще один способ регулирования порядка доступа к памяти, если запросчиков несколько. В этом варианте вместо того, чтобы разрешать конфликты, мы не будем их создавать. Конфликтов не будет, если пакеты, приходящие по UART-каналам, будут разнесены во времени. Для реализации разнесения пакетов во времени необходим механизм, который будет посылать сигналы в последовательный АЦП со скоростью передачи битов в UART-интерфейсе. С каждым таким сигналом будет выполняться аналогово-цифровое преобразование старшего бита данных, которые на тот момент хранятся в АЦП. Как только очередной бит был преобразован, он сразу же будет передаваться по UART-каналу. А контрольная сумма будет генерироваться в процессе передачи пакета.

Само разнесение во времени будет происходить следующим образом (рис. 13). Обращения к разным АЦП будут происходить поочередно с некоторым фиксированным интервалом времени между ними: сначала к первому, затем через некоторое время ко второму, потом через такое же время к третьему и т.д. Такой подход позволит избежать конфликтов при приеме пакетов во временный буфер, так как UART-посылки будут приходить в разные моменты и будет достаточно времени до следующей посылки, чтобы совершить запись. Для того, чтобы можно было использовать как можно больше датчиков, следует минимизировать интервалы между UART-посылками от разных АЦП, но если их выбрать слишком малым, то из-за неточности UART-интерфейса они могут прийти одновременно так, что

произойдет коллизия – обе посылки должны быть записаны одновременно, но это не предусмотрено в аппаратуре. Достоверно можно сказать, что по времени начало UART-посылки может прийти с ошибкой не большей чем 1/16 битового интервала, поэтому интервалы между посылками можно сделать равной 2/16 битового интервала.

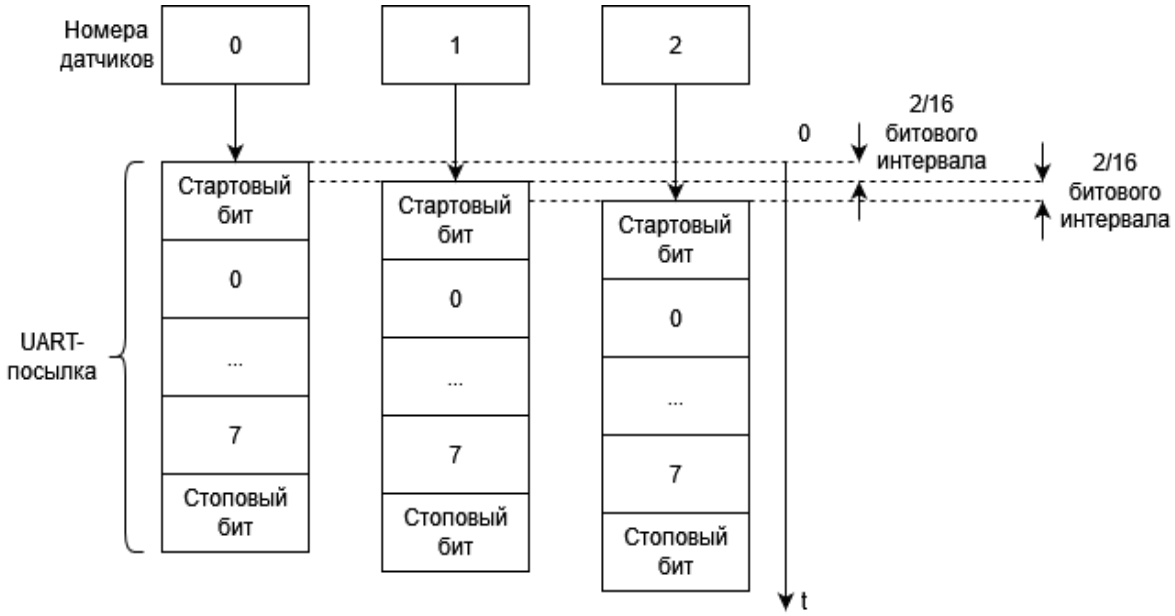



*Рисунок 13. Разнесение UART-посылок во времени.*

### 6.2.4. Блок проверки контрольной суммы

Как уже указывалось раньше, достоверность принятых данных должна определяться проверкой контрольной суммы, причем для каждого канала может использоваться свой метод генерации контрольного кода, но интерфейс модуля проверки CRC-кода одинаковый. Опишем его входные и выходные сигналы (табл.3).

*Таблица 3 - Интерфейс модуля проверки контрольной суммы*

| Имя сигнала | Разрядность | Тип | Функция |
|---|---|---|---|
| data | 8 | вход | Прием данных из UART-канала |
| load | 1 | вход | Разрешение записи данных в модуль |
| init | 1 | вход | Переводит модуль в начальное состояние |
| EoCh | 1 | выход | Сигнал, сообщающий о том, что пакет, преданный по UART-каналу, был проверен. Для генерации этого сигнала в |

| | | | |
|---|---|---|---|
| | | | этом модуле должна быть машина состояний, количество состояний которой должно быть равно количеству передаваемых байт. |
| ok_rx | 1 | выход | Сигнал, сообщающий о том, что пришел пакет с ошибкой (сигнал равен 0) или без (сигнал равен 1). Этот сигнал и записывается в биты целостности. |

### 6.2.5. Формат данных, передаваемых в канал DMA

По окончании проверки всех пакетов и записи их во временный буфер необходимо передать данные в DMA-канал. Они должны состоять из следующих частей: параметры состояния объекта управления (все, что находится в буфере), биты целостности (на схеме ok_rx) и контрольная сумма единого пакета состояния (контрольная сумма для всего, что находится в буфере, и битов целостности). Но перед этим биты целостности должны быть байт-ориентированы, дополнив их единицами до кратного числа байтов. (Пример. Если биты целостности имели бы следующие значения 11111011 110111, то записанные значения имели бы вид 11111011 00110111. Подчеркнутые нули – добавленные.) Контрольная сумма должна вычисляться пока передаются данные в канал DMA, причем каждый байт данных должен обрабатываться за 1 такт в модуле Gen_CRC. Порядок выдачи данных в канал DMA определяется коммутатором: сначала данные из буфера, потом биты целостности и последние – байты контрольной суммы единого пакета состояния.

### 6.2.6. Генератор контрольной суммы единого пакета состояния

Опишем интерфейс генератора контрольной суммы единого пакета состояния (табл.4).

*Таблица 4 - Интерфейс генератора контрольной суммы ЕПС*

| Имя сигнала | Разрядность | Тип | Функция |
|---|---|---|---|
| data | 8 | вход | Прием данных из буфера и битов целостности |
| load | 1 | вход | Разрешение записи данных в модуль |
| init | 1 | вход | Переводит модуль в начальное состояние |
| CRC_byte_number | 1 | вход | Указывает, какой из байтов контрольной суммы должен быть записан в DMA-канал |
| CRC | 8 | выход | Сигнал, содержащий CRC-код единого пакета состояния |

### 6.2.7. Интерфейс передачи данных в DMA-канал

Данные должны передаваться в DMA-канал посредством стандартизованной шины Avalon-ST. Опишем этот интерфейс (табл.5).

*Таблица 5 - Интерфейс взаимодействия DMA и внешней аппаратуры*

| Имя сигнала | Разрядность | Тип | Функция |
|---|---|---|---|
| sink_ready | 1 | Вход | Сигнал, сообщающий о том, что DMA-канал готов к приему данных |
| source_valid | 1 | Выход | Сигнал, сообщающий о том, что источник готов к передаче данных |
| source_data | 8 | Выход | Данные |
| source_eop | 1 | Выход | Сигнал конца передачи пакета данных |
| source_sop | 1 | Выход | Сигнал начала передачи пакета данных |

Данные могут передаваться только тогда, когда оба сигнала source_valid и sink_ready установлены в 1, поэтому данные следует выдавать только, когда оба этих сигнала равны 1.

### 6.2.8. Возможные дополнения к структуре измерительной подсистемы

UART-интерфейс имеет достаточно низкую скорость передачи данных и к тому же является асинхронным. Это может привести к неконтролируемым задержкам в канале связи до порядка 10мкс (1/(9600бод*16)) из-за того, что

определить начало посылки можно с точностью не больше 1/16 битового интервала. В связи с этим в высоко динамичных ЦСУ, предназначенных для управления крупными объектами, стоит UART-интерфейс заменить на интерфейс с шиной не меньшей длины, но более точный и быстрый.

В ЦСУ для малогабаритных объектов управления можно заменить UART-интерфейс и интерфейс датчика на один интерфейс – датчика и формирователь временных меток внести в концентратор. Это позволит уменьшить количество проводов, но сделает невозможным разнесение во времени пакетов, несущих параметры состояния.

Если применяется разнесение пакетов во времени, то можно контролировать скорость выдачи битов из АЦП датчиков. Например, в случае, когда датчики имеют интерфейс SPI, в котором, управляя положением фронтов сигнала CLK, можно регулировать скорость преобразования.

Для ЦСУ с крупными объектами управления, где формирователь временных меток внесен в концентратор, и с разнесением пакетов во времени, интервал следования между пакетами должен быть увеличен до 4/16, так как задержка может быть не только при приеме пакета, но и при передаче команды фиксации.

Можно было бы предложить, чтобы служба единого времени и формирователь временных меток были для каждого датчика свои. Но такой подход не только не принесет заметных улучшений, но и потребует синхронизации генераторов частоты для каждого датчика, потому что их частоты должны быть полностью одинаковыми для синхронной работы ЦСУ. Однако, не бывает генераторов с абсолютно равными частотами, всегда есть погрешность.

### 6.3. Реализация функции записи ЕПС в память расчетной подсистемы в ПЛИС-системе на кристалле

После того, как все данные были записаны во временный буфер и проверены на целостность, их необходимо передать в память расчетной подсистемы, где по их результатам будут вычислены параметры управления объектом. Эту функцию можно реализовать с помощью общего адресного пространства или через DMA-канал. Рассмотрим пример реализации интерфейса ЕПС с процессорным ядром через DMA-канал в вычислителе, построенном как ПЛИС-система на кристалле. Для построения абонентской части канала будем использовать встроенный в среду разработки Quartus II инструмент Platform Designer. Он генерирует готовые модули, которые реализуются логикой ПЛИС и их можно интегрировать в свой проект. Для примера перечислим модули, которые будут использоваться для записи ЕПС в память расчетной подсистемы и в самой расчетной подсистеме. Это, в первую очередь, процессор Nios II, который в целом и будет выполнять роль вычислителя. Затем модули Modular Scatter-Gather DMA (mSGDMA), который выполняет функции DMA-канала, и On Chip Memory (RAM or ROM) Intel FPGA IP, представляющий собой память RAM на кристалле ПЛИС. И еще несколько вспомогательных и необязательных модулей: PIO (Parallel I/O) Intel FPGA IP – порты ввода/вывода, JTAG UART (это IP-ядро позволяет совершать отладку исполняемой программы в процессоре, а также выводить сообщения в консоли, например с помощью вызова функции printf) и System ID Peripheral (этот IP-модуль позволяет задать уникальный идентификатор (ID) для процессора в нашем проекте, чтобы случайным образом не прошить не то, что требуется, в другую плату). Все вышеперечисленные модули являются внешними устройствами по отношению к процессору, к которым он может обращаться через регистры, подключенные в общее адресное пространство.

### 6.3.1. Применение инструментов для создания и прошивки процессора

Для того, чтобы создать функционирующий процессор, необходимо сначала сконструировать его в качестве модуля в Platform Designer, а затем прошить с помощью приложения Eclipse, тоже встроенным в Quartus II.

Опишем этот процесс в виде алгоритма (см. приложение «алгоритм создания процессорной системы»).

### 6.3.2. Принцип работы процессора и внешних устройств

Есть два способа работы с процессором: использовать функции операционной системы или просто написать команды, которые будут исполняться процессором. Эти команды могут быть написаны, например, на языке Си, что и было сделано в этом проекте. Существуют отдельные команды, называемые макросами, с помощью которых можно обращаться к внешним устройствам. Каждое устройство имеет свои регистры, отображённые в адресное пространство, и для управления этим устройством необходимо записывать определенные значения в его регистры с помощью макросов. Например, у модуля PIO есть только один регистр и в него можно записать значение, которое требуется использовать за пределами процессора.

Для обращения к регистрам IP-модулей у каждого есть свой уникальный адрес. Он указывается в байтах и состоит из двух частей: базового адреса и смещения. Базовый адрес указывает на устройство и его порт, а смещение на – регистр внутри этого устройства или порта.

Теперь опишем формат макросов. Есть всего два типа команды: запись в регистр и чтение из него.

IORD_<название устройства>_<имя регистра> (<базовый адрес устройства>)

IOWR_<название устройства>_<имя регистра> (<базовый адрес устройства>, <данные>)

IORD и IOWR в начале команды означают операции чтения и записи соответственно. Далее следует название устройства, к которому необходимо обратиться. Также сюда может входить и название порта, если их несколько. И в конце написания макроса идет имя регистра данного устройства или порта. В качестве аргументов таких команд выступают базовый адрес устройства или

порта и, если это запись, то еще и данные, которые должны быть записаны. Таким образом, адрес регистра в макросе записывается с помощью базового адреса и имени регистра. Приведем пример.

IOWR_ALTERA_MSGDMA_DESCRIPTOR_LENGTH(MSGDMA_0_DESCRIPTOR_SLAVE_BASE, 0x6);

Эта команда записи в порт дескрипторов (DESCRIPTOR) модуля DMA-канала (ALTERA_MSGDMA). Базовый адрес этого порта хранится в константе MSGDMA_0_DESCRIPTOR_SLAVE_BASE и равен 8240. А смещение определяется именем регистра (LENGTH) и равно 8. Так как разрядность процессора равна 32, то каждый регистр состоит из 4 байт, поэтому все адреса должны быть кратны 4. И число 0x6 предназначено для записи в регистр в качестве данных.

### 6.3.3. Описание алгоритма программы управления DMA-каналом

Опишем один из алгоритмов управления DMA-каналом. Перед запуском DMA-обмена необходимо настроить канал. Сначала, выдаётся команда, которая останавливает передачу в канале. Затем, устанавливаются параметры обмена. В нашем случае их всего 3:

1. Адрес чтения. Указывает на элемент в области памяти, который будет прочитан первым (так как данные поступают потоком, а не считываются из памяти, то любое значение, записанное в этот регистр, будет игнорироваться).
2. Адрес записи. Указывает на ячейку в области памяти, в которую будет записан первый элемент. Причем в четную и нечетную передачи будем чередовать эти адреса для того, чтобы реализовать функцию полукарманов. Она нужна, чтобы, если пакет придет с ошибками, то можно было бы использовать параметры состояния из предыдущего цикла регулирования.

3. Размер пакета. Указывает, количество байт, которое содержится в принимаемом пакете.

После того, как параметры были выбраны, требуется их зафиксировать еще одной командой.

Следующим этапом будет установка готовности передачи данных, и передача начнется, когда данные будут актуальными. Но при этом следует указать, что, если будут возникать ошибки, то передачу следует остановить. Затем необходимо сообщить процессору, что передача параметров завершена. Это можно сделать с помощью двух механизмов: прерывания или программного опроса состояния устройства DMA. Так как программный опрос реализовать гораздо проще, поэтому он и был использован. Но для этого также требуется запретить прерывания.

Как только передача была выполнена, описанная процедура повторяется снова.

## 7. Вычисление параметров управления

Вычисление параметров управления начинается сразу же после того, как при очередном опросе DMA стало известно, что весь ЕПС уже записан в память вычислителя. Сначала необходимо проверить контрольную сумму единого пакета состояния, если проверка прошла успешно, то этот пакет можно использовать для вычисление параметров управления, иначе придется использовать предыдущий или, если это критично, то остановить работу ЦСУ. Затем, если все в порядке, нужно проанализировать биты целостности. Если один из них равен нулю, значит пакет состояния от соответствующего датчика пришел с ошибкой. Эту информацию можно использовать по-разному в зависимости от ситуации: остановить работу всей ЦСУ, остановить работу только определенного контура регулирования или продолжить вычисления параметров управления, но использовать для этого старые данные. И уже после всех проверок можно проводить вычисления, для которых необходимы

не только параметры состояния, но и уставки от директивной системы верхнего уровня для задания миссии объекта управления.

Дополнительно стоит отметить характер вычислений в ЦСУ с синхронной активной периферией. Благодаря тому, что операции ввода/вывода в ЦСУ вынесены из-под программного управления и как следствие данные в расчетную подсистему поступают в виде ЕПС, программный алгоритм, который заложен в вычислителе, будет максимально соответствовать аналитическому описанию задачи управления объектом, так как возможна будет организация вычислений без прерываний.

После того, как параметры управления были получены, они записываются в память вычислителя вместе с контрольным кодом и аналогично выводятся в исполнительную подсистему через DMA-канал.

## 8. Исполнительная подсистема

Выбранный способ анализа информационных процессов в ЦСУ применим и к другим подсистемам. Так ЦСУ с активной синхронной периферией идеально подходят для согласованного управления узлами объекта управления, когда несколько актуаторов формируют общее – результирующее управляющее воздействие. Рассмотрим, как пример, управление платформой, которая передвигается на плоскости. У нее есть две степени свободы: ехать вперед-назад и влево-вправо. Платформе нужно проехать по кратчайшему расстоянию от одной точки до другой и если один из параметров управления начнет отрабатываться раньше другого, то пройденное расстояние будет больше прямого, что можно увидеть на рисунке 14.

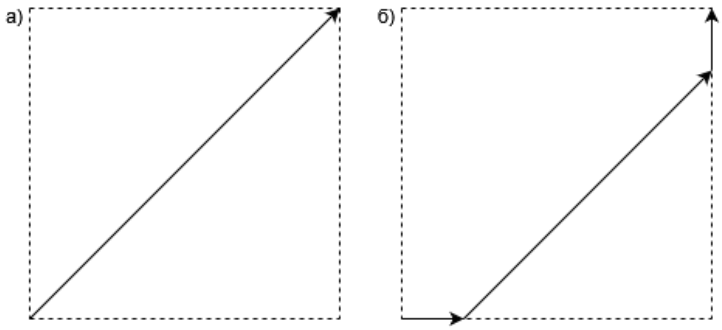



*Рисунок 14. Синхронная отработка параметров. а) Случай одновременной отработки параметров, б) Случай, когда отработка параметра, отвечающего за вертикальное движение, началась позже.*

Однако, одновременная выдача параметров управления актуаторам может привести к негативным последствиям, в частности к перегрузке системы питания из-за пикового энергопотребления. Это особенно заметно, если для выдачи управляющих воздействий применять повсеместно используемую широтную модуляцию – ШИМ. Многоканальный генератор ШИМ-сигналов строится с единым опорным счетчиком, начало сигналов формируется от его нулевого значения, а длительность задается индивидуальными уставками. В таком генераторе передние фронты ШИМ-сигналов появляются одновременно во всех каналах (рис. 15а)

Проблему пикового энергопотребления можно избежать, если фронты ШИМ-сигналов немного разнести во времени, выдавая параметры управления не одновременно. Тем не менее, нужно следить, что бы фронты были разнесены не более чем на меньший период ШИМ-сигнала, чтобы отработка параметров оставалась все еще синхронной. К примеру, можно разделить период на количество частей, равное количеству исполнительных устройств минус один, и взять этот временной интервал в качестве задержки относительно соседнего канала (рис.15б). Или же, чтобы минимизировать электромагнитные помехи, возникающие при передаче ШИМ-сигнала, можно размыть спектр этих помех, разнеся фронты псевдослучайным образом.

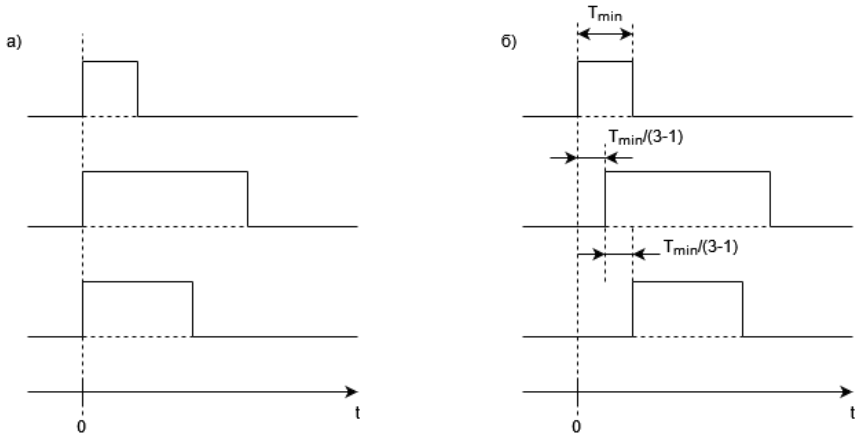



*Рисунок 15. а) Одновременное появление фронтов ШИМ-сигналов, б) Разнесение фронтов во времени.*

## 9. Реализация исполнительной подсистемы

Как и операции ввода в измерительной подсистеме, операции вывода можно вынести из расчетной подсистемы в исполнительную и реализовать ее аппаратными средствами (рис.16). Опишем последовательность действий при передаче параметров управления. Сначала параметры, хранимые в памяти, считываются из нее через DMA-канал в виде единого пакета управления (ЕПУ). Он похож на ЕПС, только содержит параметры управления и контрольный код не для всего пакета, а для каждого исполнительного устройства. ЕПУ предается в устройство – аналог концентратора, но с противоположной функцией. Там части ЕПУ, относящиеся к определенным актуаторам, записываются в соответствующие регистры. Так как разрядность параметров управления для каждого исполнителя фиксирована и известна, то с помощью счетчика и дешифратора можно части ЕПУ записать в соответствующие регистры. Затем проверяется контрольный код. После, в определенные моменты времени, определяемые фазами начала отработки уставок, которые будут формироваться в формирователе временных меток для синхронизации с остальными подсистемами ЦСУ, значения параметров из регистров передаются в опорный счетчик для формирования ШИМ-сигнала.

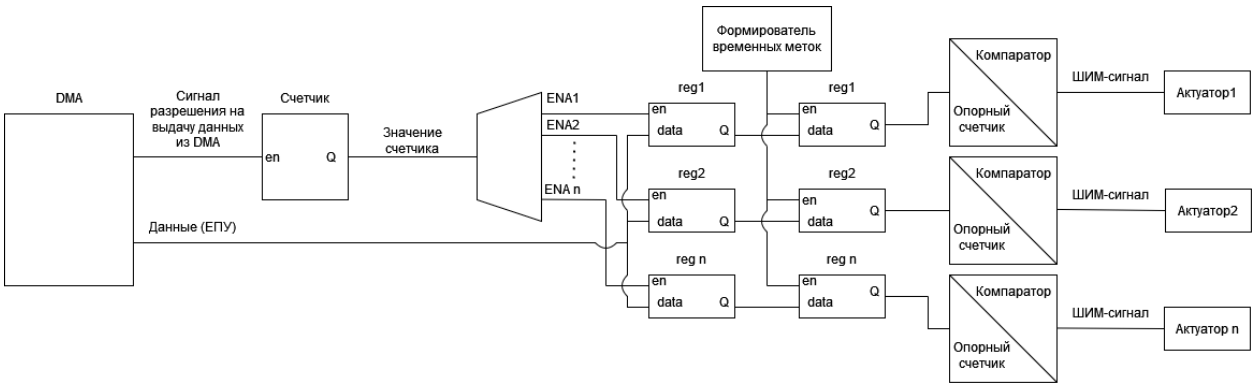



*Рисунок 16. Исполнительная подсистема.*

## 10.Хранение и анализ параметров состояния и управления

Из-за того, что набор параметров состояния фиксируется в один момент времени и набор параметров управления отрабатывается тоже одновременно, то эти структуры данных можно назвать векторами состояния и управления соответственно (рис.17).

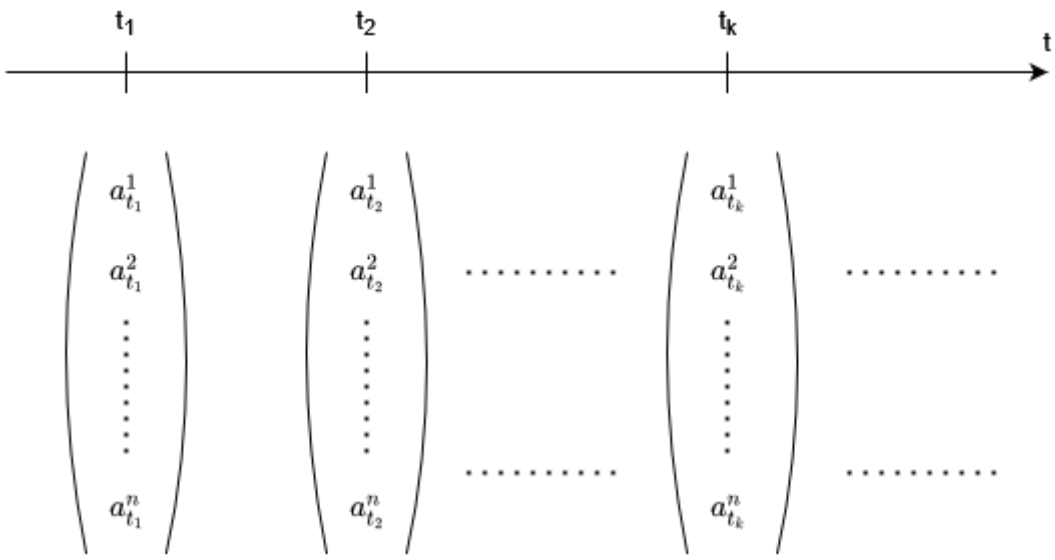



*Рисунок 17. Векторы состояния/управления.*

На выходе модуля концентратора выдается ЕПС, который в удобном формате несет вектор состояния. Аналогично ЕПУ содержит вектор управления. Их можно пересылать не только внутри ЦСУ, но и в некоторые внешние хранилища (рис.18). Это очень легко реализовать через шины Avalon-ST, по которым подключаются DMA. Получим, что вектора состояния и управления можно считывать, когда они передаются в DMA для передачи ЕПС

и из DMA для передачи ЕПУ соответственно. При этом считывание не оказывает никакого влияния на работу ЦСУ, т.е. можно сказать, что производится «прослушивание» линии передачи данных. Подобный прием реализовать на ЭВМ гораздо тяжелее. Для этого пришлось бы дополнительно создавать механизм агрегирования данных, которые к тому же приходят не одновременно.

В итоге, хранению векторов состояния и управления можно найти полезное применение. Например, использовать в качестве бортового самописца – «черного ящика», причем в ЕПС и ЕПУ можно передавать не только параметры, но и время их фиксации или начала отработки, как один из параметров. Этого уже нельзя добиться в универсальных вычислителях, так как там невозможно определить моменты фиксации параметров и выдачи уставок.

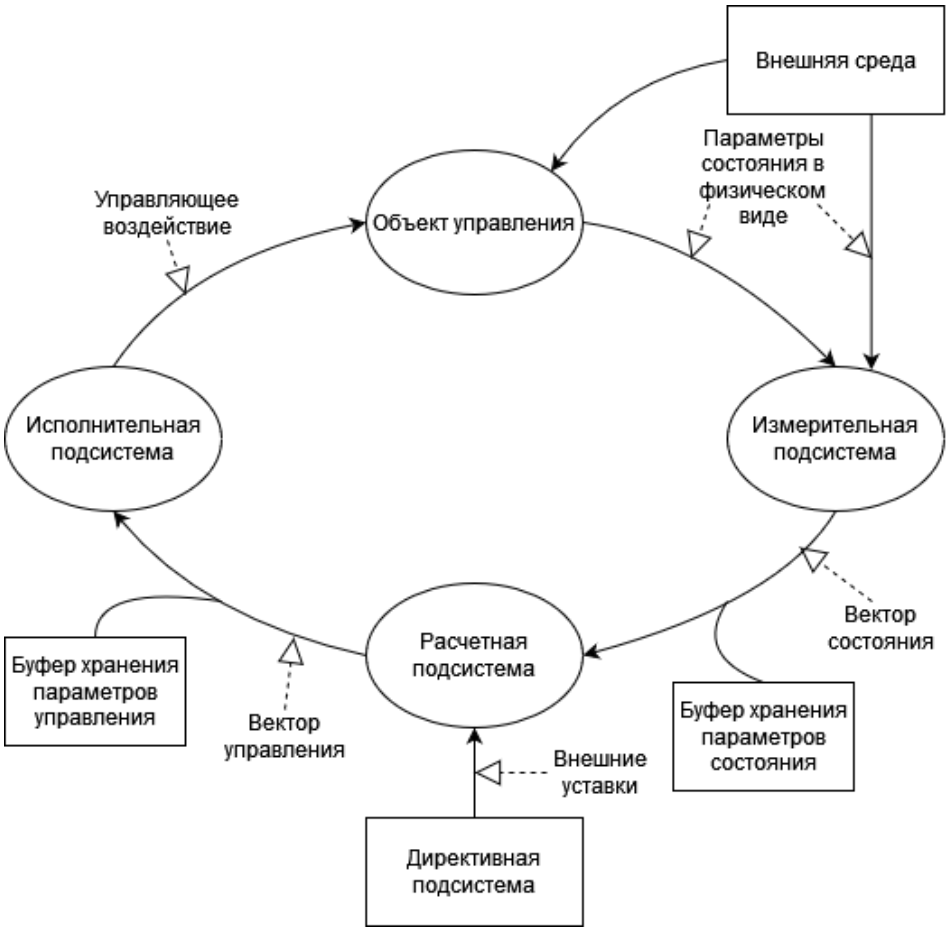



*Рисунок 18. Схема ЦСУ с хранением параметров.*

Можно пойти дальше и использовать хранящиеся параметры для создания адаптивных систем управления. Они, анализируя значения параметров состояния и управляющие воздействия, могут составить более точный алгоритм управления.

## 11.Построение модуля концентратора пакетов состояния

В ходе работы средствами ПЛИС на платформе DE2-115 с кристаллом Cyclone IV E был создан параметризованный модуль концентратора параметров состояния (см. приложение «схема концентратора»). В его состав входят модули UART-приемников, блок многоканального доступа к памяти, временный буфер, модули проверки контрольной суммы и DMA-канал. Блок генератора контрольного кода ЕПС был опущен для простоты отладки модуля концентратора и его разработка не является целевой задачей. Для управления DMA-каналом необходимо было также создать процессор. В качестве генераторов UART-пакетов были построены UART-передатчики, которые располагаются в том же кристалле, где и концентратор. Они посылают пакеты по внешнему сигналу – нажатию кнопки, содержащие значения счетчиков, которые также увеличиваются на единицу после каждого нажатия кнопки. Блок многоканального доступа к памяти реализован с логикой приоритетов обращения абонентов к временному буферу. Контрольная сумма пакетов состояния генерируется по протоколу Modbus CRC16.

Ресурсы платы, которые были использованы для реализации проекта являются логические элементы кристалла (которых имеется 114480), кнопка, а также переключатели и светодиоды для отладки и индикации.

Все модули описаны на языке Verilog за исключением DMA-канала, процессорной системы и временного буфера. Они сконструированы, как IP-модули.

Проект модуля концентратора параметризован. Перед генерацией проекта можно задать количество датчиков и размеры UART-пакетов. В

зависимости от значения этих параметров изменяется количество требуемых логических элементов для реализации концентратора.

Рассмотрим зависимость количества необходимых ресурсов от типовых значений параметров. Затрачиваемая логика на построение DMA-канала и процессорной системы почти не зависит от количества датчиков и пакетов и составляет примерно 2000 LEs. Далее на графике приведена зависимость объема логики, требуемой для реализации концентратора без DMA-канала, от количества используемых датчиков для разных размеров UART-пакетов (1 байт, 2 байта, 5 байт).

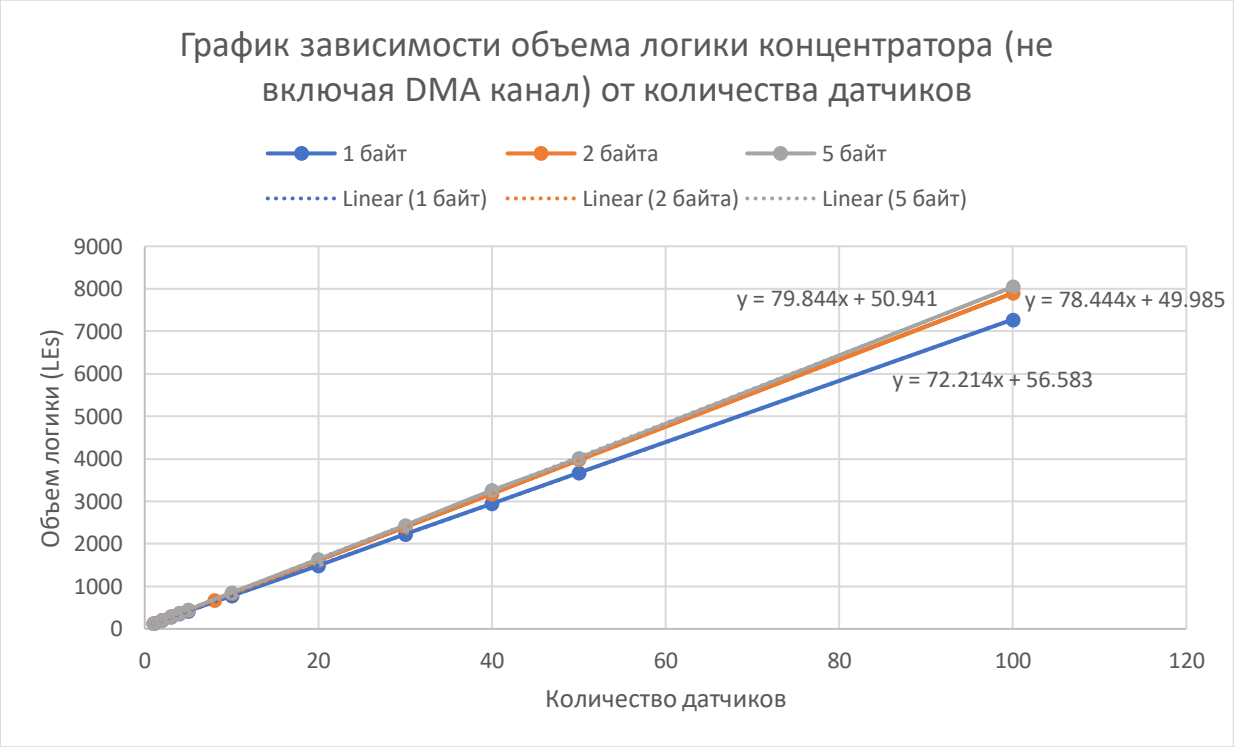



*Рисунок 19. График зависимости объема логики концентратора от количества датчиков.*

Из графика (рис.19) легко можно увидеть, что зависимость объема логики от количества датчиков линейная. Причем увеличение длины UART-пакета совершенно незначительно влияет на объем логики.

Что касается затраченного объема памяти, наибольшая его часть требуется для построения процессорной системы и DMA-канала – около 66 Кбайт. Для реализации временного буфера требуется значительно меньше.

Этот объем можно посчитать по формуле $\log_2(\text{количество датчиков}) \times \log_2(\text{длина наибольшего пакета})$, где результат приводится в байтах, а каждый логарифм округляется до целого в большую сторону.

В результате получаем, если использовать 100 датчиков и размер пакетов будет составлять 5 байт, то суммарный объем логики концентратора составит примерно 10 000 логических элементов (8,7% от общего объема логики кристалла), а объем памяти – 66Кбайт + 21байт (14% от общего объема памяти кристалла).

Проверка работоспособности концентратора проводилась в отладчике приложения Nios II Software Build Tools for Eclipse. Для этого необходимо сначала прошить плату, затем записать код программы в процессор. Потом, не отключая плату от прошивающего устройства (пользовательского компьютера), запустить отладчик. Сгенерированная система будет вести себя как при обычном исполнении, только в специальном окне отладчика можно наблюдать за состоянием регистров, переменных и памяти на каждом этапе выполнения программы. Возможность просматривать память является самым важным, так как именно там оказываются данные, которые пришли из измерительной подсистемы. В процессе отладки наблюдал, как после каждого нажатия кнопки значения в ячейках памяти увеличивалось на единицу, что указывало на то, что система работает, так как требуется.

Для считывания данных из памяти в процессор можно создать массив 8-ми битных элементов и значение указателя на первый элемент массива сделать адресом, с которого начинается запись параметров в память из DMA-канала. Таким образом, для чтения определенного параметра нужно просто обратиться к соответствующему элементу массива по его номеру.

## ЗАКЛЮЧЕНИЕ

В работе представлена модель вычислителя цифровой системы управления с активной синхронной периферией. Указаны ее состав и свойства.

Описана возможность декомпозиции информационной среды ЦСУ на три слабо связанные подсистемы, что упрощает ее структуру. К тому же это позволяет производить отладку и разработку разных подсистем независимо друг от друга. Предложены реализации этих подсистем и стык между ними средствами программируемой логики с использованием технологии Platform Designer.

Использование активной периферии и вынесение функций управления вводом/выводом из расчётной подсистемы в измерительную и исполнительную позволяют сделать расчетную подсистему универсальной для многих задач управления в технике, так как зависимость расчетной подсистемы от измерительной благодаря такому подходу значительна снизилась: больше не надо посылать различные запросы датчикам, на которые они реагируют по-разному, а только принимать данные в типовом формате. Вычислитель никак не зависит от способа измерения параметров системы, так как оперирует только с чистыми данными, которые не содержат информацию об интерфейсе между измерительной и расчетной подсистемами, поэтому можно сделать вывод, что задачи, исполняемые на нем, максимально приближены к аналитическому виду.

Показано применение согласованного управления в реализации механизма синхронной фиксации параметров для измерителей, бесконфликтной передачи данных и фазированной выдачи уставок. В синхронном режиме можно разнести во времени целые операции, которые логически друг с другом не связаны, но косвенно влияют друг на друга, например посредством электромагнитных взаимодействий, из-за чего могут возникать помехи. К тому же только в синхронном режиме возможна реализация «мгновенного снимка», а также согласованное управление разными частями объекта управления, которые отвечают разным степеням свободы. Таким образом, синхронные режимы управления периферией

повышают согласованность работы различных частей ЦСУ и, как следствие, качества управления.

Построен модуль приема данных и передачи значений параметров в вычислитель, реализованный в виде модуля концентратора пакетов состояния.

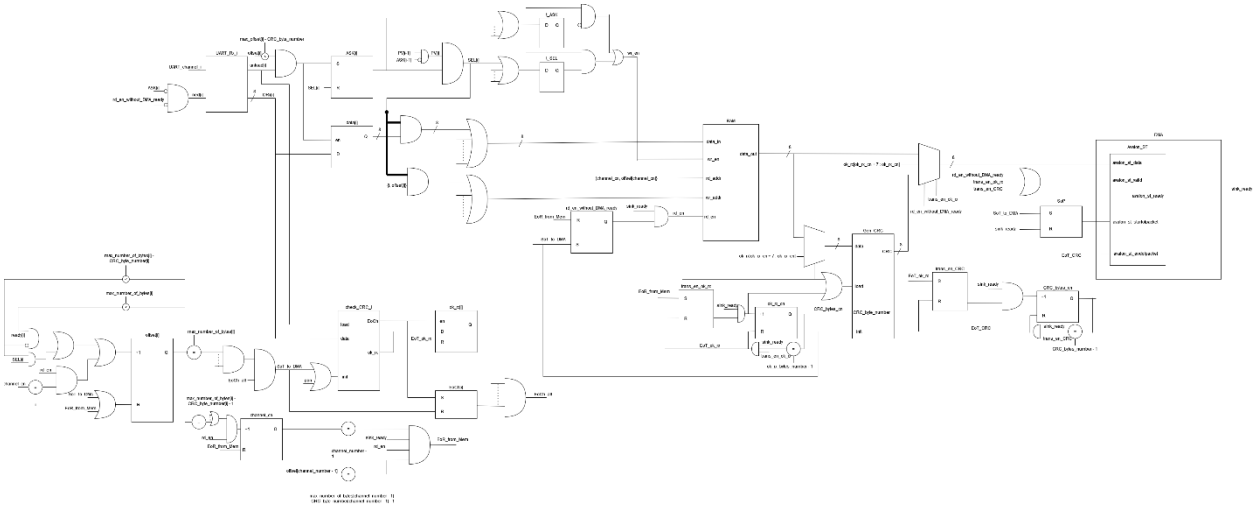
Так же был предложен способ многоканального доступа к памяти. Такая схема позволяет коммутировать несколько различных запросчиков к памяти, что так же делает эту коммутацию универсальной для любых абонентов.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. Холопов Ю. А., Нгуен Ван Хиеу. Реализация контроллера активной синхронной измерительной подсистемы цифровой системы управления // Информационные технологии. – 2019. - №19. – С.733-737.
2. Преображенский Н. Б., Холопов Ю. А., Гаврилов Д. А. Эффективная реализация и контроль интервалов времени в специализированных цифровых системах. Журнал радиоэлектроники [электронный журнал]. 2019. № 8. Режим доступа: http://jre.cplire.ru/jre/aug19/12/text.pdf DOI 10.30898/1684-1719.2019.8.12.
3. Холопов Ю.А., Ле Ба Чунг, Нгуен Тхань Чунг. Согласованная информационная среда для высокодинамичной системы управления // Журнал «Мехатроника, автоматизация, управление». – 12/2016. – №12(17). С. 816-820. – ISSN 1684-6427.
4. Смирнов Д.С., Дейнека И.Г., Алейник А.С., Шарков И.А. Основы разработки встраиваемых систем на ПЛИС с использованием процессора NIOS® II: Учебное пособие. - Санкт-Петербург: Университет ИТМО, 2019. - 95 с. - экз.
5. http://www.terasic.com.tw/attachment/archive/502/DE2_115_User_manual.pdf
6. Avalon Interface Specification
7. Embedded Peripherals IP User Guide
8. Nios II Classic Software Developer’s Handbook
9. https://habr.com/ru/post/500016/ (версия на момент 15.03.2021)

ПРИЛОЖЕНИЕ

## Схема концентратора



ПРИЛОЖЕНИЕ

## Алгоритм создания процессорной системы

1. Создание проекта в Quartus.

2. Работа в Platform Designer. Проектирование процессора и внешних устройств.

2.1. Запуск Platform Designer. Tools/ Platform Designer.

2.2. Добавление процессора. В IP Catalog выбираем Processor and Peripherals/Embedded Processors/Nios ii Processor.

2.3. Настройка процессора. Выбираем версию Nios ii/e во вкладке main. Во вкладке Vectors выбираем память процессора (можно не менять, но после добавления памяти нужно ее указать) в разделах «Reset vector memory» и «Exception vector memory». Во вкладке JTAG Debug установить галочку рядом с Include JTAG Debug. Нажать finish.

2.4. Создание памяти. Можно создать память на кристалле, для этого нужно в IP Catalog выбрать Basic Functions/On Chip Memory/On Chip Memory (RAM or ROM) Intel FPGA IP.

2.5. Настройка памяти. Можно выбрать тип (RAM или ROM) и размер. Нажать finish.

2.6. Создание модуля mSGDMA. В IP Catalog выбираем Basic Functions/DMA/Modular Scatter-Gather DMA Intel FPGA IP.

2.7. Настройка параметров mSGDMA.

2.7.1. Выбрать режим Streaming to Memory-Mapped для передачи потока данных в память.

2.7.2. Data Width установить в 8 бит.

2.7.3. Maximum Transfer Length установить таким образом, чтобы это значение было не меньше чем объем передаваемых данных для любого пакета.

2.7.4. Выбрать Packet Support Enable.

2.7.5. Остальное можно не менять.

2.8. Создание портов ввода/вывода. В IP Catalog выбирается Processor and Peripherals/ Peripherals/PIO (Parallel I/O) Intel FPGA IP.

2.9. Настройка портов ввода/вывода. Можно выбрать будет ли это входом или выходом, или двунаправленным портом, также размер шины данных. Нажать finish.

2.10. Создание дополнительных необязательных компонент: JTAG UART и System ID Peripheral. При их создании ничего не меняем кроме ID, который выбираем произвольно.

2.11. Соединение компонентов нажатием на точки в местах пересечения шин.

2.11.1. Выводы clk и clk_reset блока тактирования соединяем со всеми остальными блоками.

2.11.2. Шины данных (data_master) и команд (instruction_master) процессора заводим на память к порту s1.

2.11.3. Так же шину данных подключаем к каждому блоку.

2.11.4. Завести порт mm_write (данные, пересылаемые в память из DMA) в память.

2.11.5. К портам csr и descriptor_slave модуля mSGDMA подключить порт data_master процессора.

2.11.6. Подключить прерывание (csr_irq) модуля mSGDMA к порту процессора, принимающего сигналы прерывания (irq).

2.11.7. Для порта st_sink модуля mSGDMA в колонке Export указать название для подключения собственного модуля (не IP).

2.11.8. Порт блока JTAG UART, отвечающий за прерывание (irq), соединяем с портом процессора, принимающего сигналы прерывания (irq).

2.12. Распределение адресов компонент. Это можно сделать автоматически: System/Assign Base Address.

2.13. Нажать Generate HDL.

2.14. Добавить созданный модуль в проект (добавить файл с расширением qip) и соединить его с другими модулями.

2.15. Скомпилировать и прошить плату sof файлом.

3. Работа в Eclipse. Создание прошивки процессора.

3.1. Запуск Eclipse. Tools/Nios II Software Build Tools for Eclipse.

3.2. Выбор папки для размещения.

3.3. Создание проекта в Eclipse. File/New/Nios II Application and BSP from Template.

3.4. В появившемся окне выбираем файл, который был сконфигурируем в quartus, с расширением sopcinfo (внимательно выбираем файл не путая проекты). Называем проект. Выбираем шаблон (например Blank Project, который будет пустым и нужно самому создавать С-файлы). После создания должны появиться два проекта: «название_bsp» отвечает за драйверы и библеотеки, «название» - за код, который будет исполнять процессор.

3.5. Налаживание взаимодействия с консолью Eclipse во время отладки. ПКМ по «название_bsp»/Nios II/BSP Editor. В появившемся окне во вкладке Main напротив stdin, stdout, stderr выбираем jtag_uart_0. Generate.

3.6. Компиляция проекта «название_bsp». ПКМ по «название_bsp»/Build Project.

3.7. Создание файла для кода. ПКМ по «название»/New/Source File. В появившемся окне выбираем имя файла с расширением .c и шаблон (None - пустой). Finish. Пишем код.

3.8. Компиляция проекта «название». ПКМ по «название»/ Build Project.

3.9. Прошивка. ПКМ по «название»/Run As/3 Nios II Hardware. Переходим во вкладку Target Connection. Если не нашлись устройства (ПЛИС, который должен быть включен, и процессор), то нажимаем Refresh Connection. Если адреса не сходятся, то нужно

перепрограммировать ПЛИС. Если причина в другом, то ее устраняем и заново генерируем BSP следующим образом: ПКМ по «название_bsp»/Nios II/Generate BSP и «название_bsp»/Build Project. Если все в порядке, то нажимаем Run. Готово.