УДК 621.3.049

РАЗРАБОТКА И ИССЛЕДОВАНИЕ I2C/SMBUS VIP БЛОКА

Кураедов Вадим Иванович
Магистр кафедры проектирования
и конструирования интегральных микросхем,
г. Зеленоград

АННОТАЦИЯ

В последние годы современные цифровые технологии становятся непременным элементом повседневной жизни не только профессиональных сообществ, ни и всего общества в целом. Расширение области применения цифровых устройств и технологий и усложнение решаемых ими задач обуславливает, в свою очередь, непрерывный рост сложности интегральных схем (ИС), являющихся аппаратной платформой новых цифровых решений. При этом давление рынка требует от разработчиков интегральных схем сохранения времени разработки все более сложных ИС при одновременном сохранении качества разрабатываемой продукции. В этих условиях функциональная верификация разрабатываемых устройств становится краеугольным камнем всего процесса разработки, а ее эффективность фактором, определяющим скорость разработки ИС.

Целью работы является демонстрация потенциала наиболее прогрессивной на сегодняшний день методологии функциональной верификации – UVM – позволяющей резко ускорить процесс функциональной верификации современных систем на кристалле. В работе с применением методологии UVM разработан верификационный IP блок, предназначенный для проверки, разрабатываемой ИС на соответствие требованиям протокола I2C/SMBus. Разработанный верификационный IP блок поддерживает следующий функционал:

1. Выполнение основных функций I2C: инициализация и завершение передачи данных; работа в режиме 7 – битной и 10 — битной адресации; подтверждение приема;

2. Работа как в режиме Master, так и в режиме Slave;

3. Поддержание протоколов SMBus: Quick_Command, Send_Byte, Receive_Byte, Write_Byte, Write_Word, Read_Byte, Read_Word, Process_Call, Block_Write, Block_Read, Write_32, Read_32, Write_64, Read_64.

ABSTRACT

In recent years, modern digital technologies have become an indispensable element of the daily life of not only professional communities, but society as a whole. The expansion of the field of application of digital devices and technologies and the complication of the tasks they solve causes, in turn, the continuous growth of the complexity of integrated circuits (ICs), which are the hardware platform of new digital solutions. At the same time, market pressure requires integrated circuit designers to save the development time of increasingly complex ICs while maintaining the quality of the products being developed. Under these conditions, functional verification of the devices being developed becomes the cornerstone of the entire development process, and its efficiency is a factor that determines the speed of IC development.

The aim of this work is to demonstrate the potential of the most progressive functional verification methodology today - UVM - which allows to dramatically accelerate the process of functional verification of modern systems on a chip. In the work using the UVM methodology, a verification IP block has been developed, designed to check the developed IS for compliance with the requirements of the I2C/SMBus protocol. The developed IP verification block supports the following functionality:

1. Performing the main functions of I2C: initialization and completion of data transfer; work in 7 - bit and 10 - bit addressing mode; confirmation of receipt;

2. Work in both Master and Slave modes;

3. Supports SMBus protocols: Quick_Command, Send_Byte, Receive_Byte, Write_Byte, Write_Word, Read_Byte, Read_Word, Process_Call, Block_Write, Block_Read, Write_32, Read_32, Write_64, Read_64.

Ключевые слова: верификация; UVM; VIP; проектирование; протоколы.

Keywords: verification; UVM; VIP; design; protocols.

ВВЕДЕНИЕ

Для проверки, что заказная СБИС удовлетворяет всем функциональным требованиям требуется порядка 70% от всего времени проектирования. И как бы не хотелось этого признавать, однако с ростом сложности и размера разрабатываемого устройства данная ситуация только ухудшается. Для того, чтобы решить фундаментальную проблему обеспечения корректности микропроцессоров происходит применение разнообразных средств функциональной верификации. Если ошибки возникают где-то в программе, то их достаточно легко исправить, однако конструктивные и производственные дефекты, обнаруженные в интегральных схемах, не поддаются такому простому устранению [1]. Возможности средств верификации заметно отстают от возможностей систем проектирования и технологий изготовления. В связи с этим, создание новых

средств является столь актуальной задачей. Блоки верификации IP (VIP) добавляются в тестовое окружение для проверки работоспособности протоколов и интерфейсов, как дискретно, так и в комбинации. Как бы не совершенствовались системы автоматизированного проектирования (САПР), генераторы тестовых последовательностей и различные методики анализа разрабатываемых схем, тем не менее верификация была и остается самым узким местом процесса проектирования. Эта работа посвящена разработке специального верификационного блока, который значительно упростит функциональную проверку.

ПРОЦЕСС РАЗРАБОТКИ I2C/SMBUS VIP

Работа над VIP началась с описания базовой структуры всех необходимых файлов: INTERFACE, SEQUENCER, AGENT, DRIVER, MONITOR, SEQ_LIB, TRANSACTION. На начальном этапе была реализована работа VIP по фиксированию на шине SDA состояния START CONDITION, приему 7 бит адреса, бита R/W, показывающего режим работы, и сравнение этого адреса с адресом SLAVE. Помимо этого, добавлено фиксирование состояния STOP CONDITION. На следующем этапе планировалось реализовать возвращение ведомым ACK или NACK. С архитектурой VIP окружения можно ознакомиться на рисунке 1.

Для отладки работоспособности написанного функционала VIP, на ранних этапах создавался тест, который имитировал отправку данных на шины SDA и SCL, так как подключать для этих целей RTL на данный момент не представлялось возможным, по причине отправки последним большого объема неконтролируемых данных. Как и планировалось, после отправки MASTER-ом приема 8 бит на шину SDA (адрес + R/W), добавился механизм сравнения полученного адреса с адресом SLAVE и возвращению последним ACK или NACK. После этого, в случае совпадения адреса (ACK), происходит неограниченный прием байт данных с ответом на шине SDA 9 битом ACK, как подтверждение того, что байт данных принят. Прием данных будет производиться до тех пор, пока не придет STOP CONDITION или REPEAT START (SR).

Как и планировалось, после отправки MASTER-ом приема 8 бит на шину SDA (адрес + R/W), добавился механизм сравнения полученного адреса с адресом SLAVE и возвращению последним ACK или NACK. После этого, в случае совпадения адреса (ACK), происходит неограниченный прием байт данных с ответом на шине SDA 9 битом ACK, как подтверждение того, что байт данных принят. Прием данных будет производиться до тех пор, пока не придет STOP CONDITION или REPEAT START (SR).

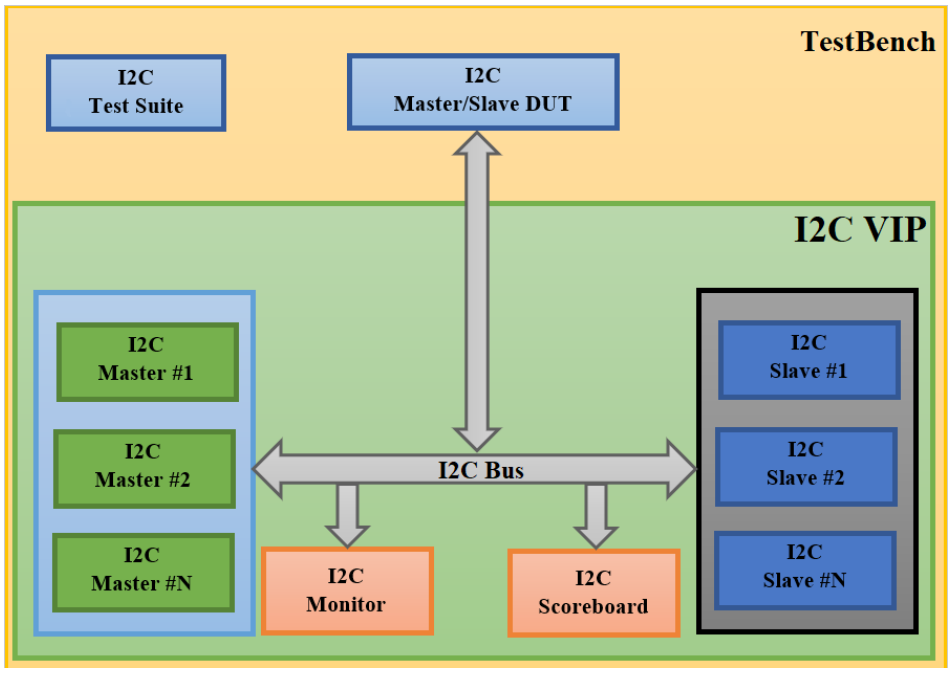

Рис.1. Архитектура VIP окружения

Таким образом, работа в режиме 7-битной адресации на своем начальном этапе была отлажена. Известно, что I2C для увеличения количества возможных подключаемых устройств поддерживает 10-битную адресацию. В связи с этим, приступил к реализации данной функции в создаваемом VIP. Для этого требовалось ввести исключение для комбинации адресов типа 11110XX и при совпадении приходящего адреса с этим шаблоном, принимать еще один байт, для получения полного 10-битного адреса. В результате данный тип адресации успешно добавился в список поддерживаемых функций I2C VIP.

К концу первой фазы разработки, блоком поддерживались следующие функции [2]:

1.Инициализация и завершение передачи данных (START и STOP);

2.7-разрядная адресация:

a)прием 7 бит адреса SLAVE;

b)считывание 8-го бита режима работы R/W;

c)сравнение принятого адреса с SLAVE_DEFAULT;

d)прием данных будет производиться до тех пор, пока не придет STOP CONDITION или REPEAT START;

e)чтение данных будет производиться до тех пор, пока не закончатся байты данных или не придет STOP.

3.10-разрядная адресация:

a)считывается два старших бита адреса и сравнивается с двумя старшими битами SLAVE_DEFAULT (10-битый адрес SLAVE);

b)считывание бита режима работы R/W;

c)считывание второго байта, в котором передается 8 младших бит адреса;

d)сравнение принятого адреса с SLAVE_DEFAULT;

f)прием данных будет производиться до тех пор, пока не придет STOP CONDITION или REPEAT START;

e)чтение данных будет производиться до тех пор, пока не закончатся байты данных или не придет STOP CONDITION.

4.Подтверждение приема (При передаче данных после каждого переданного байта выставляется ACK или NACK)

В работу VIP добавился обработчик случая, при котором сначала происходит обращение по адресу в режиме записи, а после, не передовая ни одного байта данных, сразу SR и переход в режим чтения (Рис. 2). В дальнейшем планировалось расширить функционал VIP-блока.

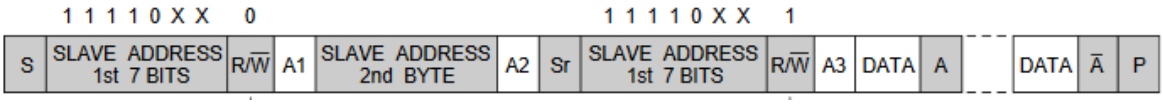

Рис. 2. MASTER приемник обращается к SLAVE передатчику с 10-битным адресом

Следующее нововведение возможность поддержки VIP нескольких SLAVE устройств. Их количество определяется в файле конфигурации. Массив всех возможных адресов жестко прописан (их генерация не рандомная), при желании его можно дополнить или изменить порядок элементов. Запись данных для каждого ведомого производится в свой собственный регистр, что позволяет избежать переписывания и создать модель поведения обмена данными между несколькими устройствами в реальных условиях. Добавился в работу VIP упущенный на этапе базовой разработки блока случай, при котором с приходом бита NACK ведущий должен формировать сигнал STOP. На тот момент была реализована работа с несколькими ведомыми только для 7-битной адресации и в дальнейшем планировалось расширить функционал VIP блока добавив в него и 10-битную адресацию.

По итогу во второй фазе I2C VIP еще умеет:

1)Поддерживать бесконечное число (ограниченное только размерностью созданного и заполненного массива) 7-битных и 10-битных SLAVE адресов;

2)Выделять память и записывать данные в свой конкретный SLAVE;

3)Понимать, когда у него 7-битная адресация, а когда 10-битная;

4)Обрабатывать всех предусмотренных стандартом форматы пакетов;

После того, как работа по написанию VIP в режиме SLAVE закончилась, то сразу приступил к разработке последнего в режиме MASTER. Первое с чего начал, это формирование на шине START CONDITION и генерации необходимых синхросигналов. После упорной работы получилось реализовать базовый функционал VIP:

1)START CONDITION

2)STOP CONDITION

3)Генерацию SCL

4)Проверку прихода ACK (режим WRITE)

5)Выставление на шину ACK (режим READ)

6)Формирование 7-битной и 10-битной адресации

7)Работа в режиме WRITE и READ

Продолжил заниматься написанием VIP в режиме MASTER. Подправил логику работы блока, добавил SR и подключил ANALYSIS_PORT, проверил, что поля транзакции успешно передаются. Добавил в том числе выбор протокола и командный код. На основании выстроенной работы MASTER, было решено внести изменения в работу SLAVE. Помимо изменения базовых функций, избавился от присутствия лишних переменных и конструкций. Сделал, чтобы хранились для последующего извлечения все записанные MASTER-ом данные. В дальнейшем планировал отладить работу VIP, по возможности внести необходимые правки для правильного функционирования блока.

Изменил структуру VIP, сделал полное разделение на файлы MASTER и SLAVE. Получилось реализовать динамическое создание MASTER_AGENT, SLAVE_AGENT и MASTER_CONFIG, SLAVE_CONFIG. Пользователь может задать из теста количество требуемых ему устройств и сконфигурировать поля каждого из них и системой корректно создадутся все MASTER и SLAVE. Данная функция была успешно проверена, все работает. В дальнейшем требовалось изменить некоторую логику работы DRIVER_SLAVE и DRIVER_MASTER, так как для такого окружения они не были написаны.

Далее изменил структуру кода, чтобы несколько SLAVE устройств корректно работали с одной шиной не создавая конфликтов. Внес изменения в алгоритм работы MASTER устройства в режиме READ. В качестве размерности принимаемого массива, берется сконфигурированный пользователем тип

протокола и как следствие количество байт предусмотренное им. Аналогичный принцип работы с разделением по типу протоколов реализован для режима WRITE.

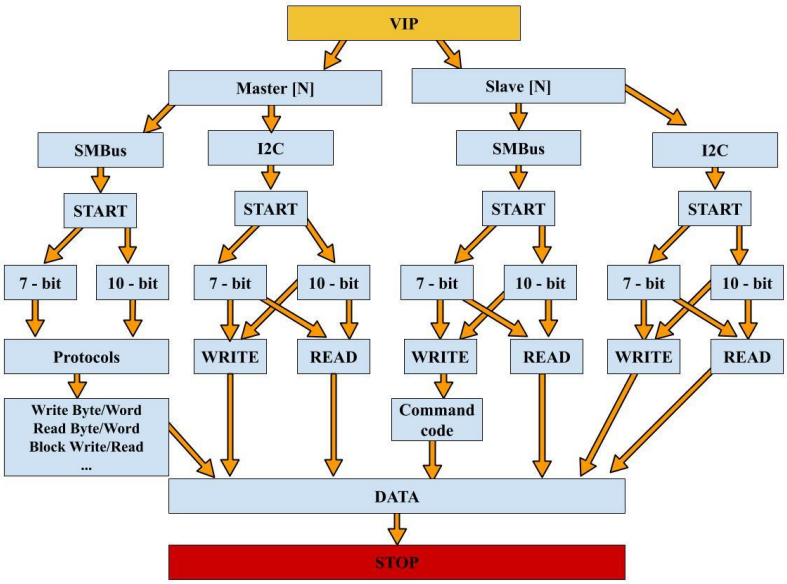

Рис.3. Схематичное представление реализованного функционала I2C VIP

Что касается работы SLAVE устройств в этих режимах, то в случае записи (WRITE) процесс продолжается до прихода SR или STOP, после чего ведомый записывает в свой внутренний массив значения для дальнейшего его изъятия. В случае чтения (READ) с точкой остановки процесса проще, так как в качестве окончания передачи достаточно получить NACK. С чем устройство прекрасно справляется.

Предыдущие тесты моего блока как MASTER в режиме чтения проводились путем передачи двух транзакций с SR, с двумя типами протоколов. Однако для пользователя удобнее одной транзакцией совершить полный цикл работы протокола. В связи с этим, в работу блока при режиме чтения добавились специальные функции, реализующие данную систему. В процессе отладки, возникало много проблем разного характера, но в итоге все типы протоколов в режиме MASTER отработали корректно.

Изначально разрабатывался I2C VIP, но с нагромождением протоколов SMBus системой уже сложно было послать обычные I2C последовательности. В связи с этим, было принято решение скорректировать работу VIP так, чтобы простой настройкой полей транзакции, можно было работать в обоих режимах.

Для наглядности всего, что реализовано на рисунке 3 представлена схема, каждый уровень в которой сейчас прокомментируем. Во главе работы стояла сама идея разработки VIP. Далее настроил поддержку его в режиме MASTER и SLAVE. Так как VIP должно быть универсальным, то добавилась настройка, по какому именно протоколу он будет работать SMBus или I2C. Когда с глобальными вопросами дело решилось, необходимо было приступить к функциональной реализации. В данном случае, все начинается со START Condition, либо его генерации, либо определения. Далее добавилась работа блока в режиме 7- или 10-битной адресации. Если адреса совпали, то определяется ветка работы в режиме чтения или записи, а в случае, SMBus MASTER-а описывается работа конкретного протокола. Так как работаем с данными, то производится их обработку. И заканчивается все генерацией, в случае MASTER-а, и определением, в случае SLAVE-а, состояния STOP.

Эта экосистема была еще раз проверена с протоколами SMBus и с различными типами посылок I2C. Исправлены мелкие недочеты и в результате все корректно отработало. Так как известно, что устройства работают на разных частотах, то помимо стандартных 100 кГц, была реализована возможность поддержки 400 кГц. Для этого пользователю необходимо лишь изменить значение параметра SPEED_MODE в файле конфигурации для каждого MASTER и SLAVE.

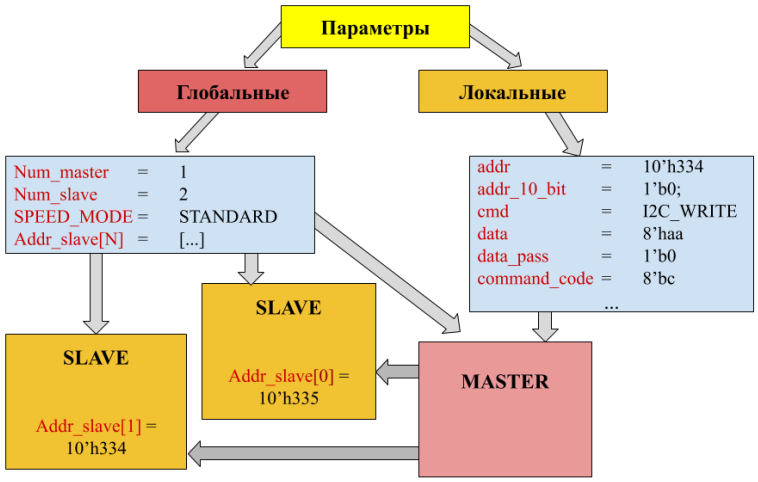

Рис.4. Структура и список конфигурационных параметров

Все параметры, используемые в VIP, можно условно разделить на глобальные и локальные (Рис. 4). Глобально указываем количество MASTER-ов и SLAVE-ов, которые будут в окружении, частоту на которой они работают и назначаем каждому из созданных ведомых устройств его уникальный адрес. Локально же, мы определяем параметры, в зависимости от которых MASTER-ом будет сформирована конкретная последовательность данных. Это адрес SLAVE-а, к которому он обратится, признак типа адресации, конкретный тип протокола, количество байт данных, непосредственно сами данные и многие другие. Различные комбинации этих параметров, позволяют по-разному сконфигурировать окружение и устройство, в частности.

Для того, чтобы убедиться, что VIP помимо формата пакета, соответствует всем предусмотренным таймингам, были замерены на обеих частотах все предусмотренные по стандарту значения временных параметров. Как видно из таблицы 1 все значения корректны, так как укладываются в заданный промежуток.

Таблица 1.

Значения временных параметров, требуемых по стандарту и реализуемых в разработанном VIP блоке

Parameters, µs	100 kHz			400 kHz		
	Standard		VIP	Standard		VIP
	min	max		min	max	
t_{BUF}	4.7	-	13	1.3	-	4.65
$t_{HD:STA}$	4.0	-	4	0.6	-	3
$t_{SU:STA}$	4.7	-	13.701	0.6	-	1.251
$t_{SU:STO}$	4.0	-	4	0.6	-	0.6
$t_{HD:DAT}$	0	-	0.3	0	-	0.3
$t_{SU:DAT}$	0.25	-	4.7	0.1	-	1.005
t_{LOW}	4.7	-	5	1.3	-	1.3
t_{HIGH}	4.0	50	5	0.6	50	1.3

На рисунке 5 можно ознакомиться со структурой файла, отвечающего за подключение всех элементов VIP. Суммарное количество строчек кода составило более 3500, а время разработки около 6,5 месяцев. Еще раз проверил работу VIP при разных конфигурационных параметрах. Как и прежде, все срабатывает

корректно, формат пакета и все тайминги совпадают.

```
package i2c_pkg;

  import uvm_pkg::*;

  `include "uvm_macros.svh"

  `include "i2c_parameters.svh"
  `include "i2c_transaction.svh"
  `include "i2c_sequencer.svh"
  `include "i2c_seq_lib.svh"
  `include "i2c_config_master.svh"
  `include "i2c_config_slave.svh"
  `include "i2c_config.svh"
  `include "i2c_monitor_master.svh"
  `include "i2c_monitor_slave.svh"
  `include "i2c_driver.svh"
  `include "i2c_driver_s.svh"
  `include "i2c_driver_m.svh"

  `include "i2c_agent_master.svh"
  `include "i2c_agent_slave.svh"
  `include "i2c_env.sv"

endpackage
```

Рис. 5. Структура файла i2c_pkg.sv

ВЕРИФИКАЦИЯ БЛОКА I2C/SMBUS

Для проверки моего VIP в работе с DUT, был сгенерирован I2C RTL с поддержкой SMBus от компании Synopsys. Благополучно подключил его в окружение. Для более корректной работы с последовательностями было принято решение добавить регистровую модель. Получив дополнительно ralf файл, сгенерировал RAL регистровую модель. Далее планировал добавить это все в окружение, использовав дополнительные файлы. Полученную регистровую модель благополучно подключил в окружение. На тестовых регистрах проверил, что запись происходит по правильным адресам.

Для начала работы с DUT требовалось получить на шине START CONDITION, как показатель того, что основные регистры заработали. Так как для этого было необходимо в определенной последовательности заполнить конкретные биты в регистрах, то данная операция была осуществлена. Таким образом, от мастера получилось добиться на шине состояния старта и 7 бит адреса. Режим WRITE и READ отработали корректно. Все отправленные данные от SLAVE к MASTER были корректно приняты последним и помещены в соответствующие регистры. Проверки были подвергнуты все форматы пакета для стандарта I2C и все протоколы стандарта SMBus: Quick_Command, Send_Byte, Receive_Byte, Write_Byte, Write_Word, Read_Byte, Read_Word, Process_Call, Block_Write, Block_Read, Write_32, Read_32, Write_64, Read_64.

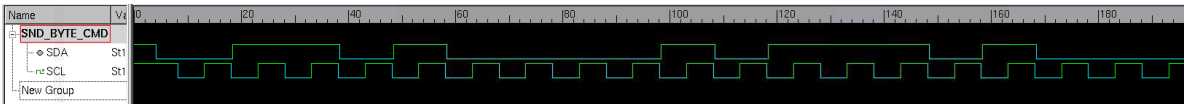

Рис.6. Обращение к SMBus SLAVE устройству с адресом 'h334 по протоколу Send Byte

Далее было принято решение заняться описанием входных воздействий для будущих тестов. В ходе повторного просмотра плана верификации, внес в него ряд изменений, связанных с исключением операций, производимых внутри блока. Результаты верификации представлены на рис. 6.

ВЫВОД

Рынок потребительской электроники требует, чтобы новые продукты развивались невероятно быстро с почти безупречным качеством. Это, в сочетании со все возрастающей сложностью устройства, создает невероятную нагрузку на инженера-верификатора SoC. Решения IP-верификации, включающие трехэтапную стратегию соответствия протоколов, анализа целостности данных и проверки производительности, необходимы для предоставления комплексного решения для проверки SoC [3].

Хотя теоретические преимущества VIP являются убедительными, практическая реализация этих преимуществ требует, адаптированного к уникальным требованиям моделирования, ускорения и формального анализа. VIP должен работать в различных средах, использующих различные языки и симуляторы, а также должен поддерживать множество протоколов. При возникновении ошибок инженер должны иметь возможность получить помощь извне. Как бы ни были важны для специалистов по проверке знания в области IP, они почти наверняка

не смогут достичь уровня эксперта, для каждого используемого IP - даже для всей команды [4]. Сегодняшние проекты включают в себя так много спецификаций, и новые добавляются с такой скоростью, что ресурсы, необходимые для достижения этой внутренней задачи, являются огромной проблемой.

В работе представлена разработка верификационного IP-блока, предназначенного для создания микросистем на основе электронных схем преобразования и обработки сигналов с использованием эффективного протокола обмена информацией I2C.

ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ

1.Y.-J. Chang, Y.-T. Lee, and T.-C. Wang. NTHU-Route 2.0: a fast and stable global router. In Proceedings of the 2008 International Conference on Computer-Aided Design (ICCAD 2008), pages 338–343, 2008.

2.I2C-bus specification and user manual Rev. 6 — 4 April 2014.

3.Parthipan, Deepak Siddharth, UVM Verification of an SPI MASTER Core, (2018). Thesis. Rochester Institute of Technology. Accessed from http://scholarworks.rit.edu/theses/9793.

4.William K. Hardware Design Verification: Simulation and Formal Method-Based Approaches. Prentice Hall PTR, 2005. 624 p.

УДК: 681.3:004
ГРНТ: 28.23.00

ИНФОРМАЦИОННО-РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА В РЕСТАВРАЦИИ

Курбанова Назакет Гаджи кызы
Кандидат технических наук, доцент,
кафедра «Медицинская и биологическая физика»,
Азербайджанский Медицинский Университет, г.Баку
Гаджиев Заур Азиз оглы
Кандидат технических наук, инженер,
«Центр организации и управления информационных систем»
Азербайджанский Медицинский Университет, г.Баку

INFORMATION AND RECOMMENDATION SYSTEM IN RESTORATION

Gurbanova Nazakat Haji
Ph.D, assistant of professor,
department "Medical and biological physics",
Azerbaijan Medical University, Baku s.
Hajiyev Zaur Aziz
Ph.D, engineer,
"Center for Organization and Management of Information Systems",
Azerbaijan Medical University, Baku s.
DOI: 10.31618/ESU.2413-9335.2020.1.76.897

АННОТАЦИЯ

Информационно-рекомендательная система построена для центра реставрации предметов. Система состоит из модулей картотека, рекомендаций, редактирования изображений. Во всех модулях информация представляется в соответствующих базах данных. Рекомендательная часть системы предлагает образцы процессов реставрации типовых случаев для текущего предмета.

ABSTRACT

An information and recommendation system built for the object restoration center. The system consists of modules card index, recommendations, image editing. In all modules, information presented in the respective databases. The advisory part of the system offers examples of restoration processes for typical cases for the current subject.

Ключевые слова: информационно-рекомендательная система, центр реставрации, база данных, банк данных, модуль.

Key words: information and recommendation system, restoration center, database, data bank, module.

Применение информационных технологий во многих областях связано с необходимостью обработки большого объема информации различного формата, возможностью разностороннего анализа предметной области, принятие решения в сложных ситуациях, объективной оценки, исключением субъективизма. Одной из таких является область искусства, в частности музейное дело. Данная область отличается тем, что музеи как хранилища большого количества экспонатов, также являются исследовательской мастерской для работы над хранимыми предметами. Центр реставрации является организацией, где хранится и ведется восстановительная работа над экспонатами. Одной из основных отличительных черт центра является