УДК 378.147:004
ББК 3973р ГСНТИ 14.85.35 Код ВАК 13.00.02

Алексеевский Петр Иванович,
ассистент кафедры информатики, информационных технологий и методики обучения информатике, Уральский государственный педагогический университет; 620075, г. Екатеринбург, ул. К. Либкнехта, 9; e-mail: u@nyuu.ru.

ОБУЧЕНИЕ СТУДЕНТОВ АРХИТЕКТУРЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА VERILOG

КЛЮЧЕВЫЕ СЛОВА: архитектура вычислительных систем; проектирование вычислительных систем; вычислительные устройства; языки описания оборудования.

АННОТАЦИЯ. В данной статье описывается необходимость рассмотрения внутреннего устройства и принципов работы функциональных блоков процессорного ядра и основных периферийных устройств в процессе обучения архитектуре вычислительных систем с учетом требований образовательных стандартов и используемых в промышленности технологий. Рассмотрены возможность и перспективы использования языка описания оборудования Verilog в процессе обучения студентов архитектуре вычислительных систем. Перечислены основные особенности языка Verilog, имеющие значение для целей обучения. Рассмотрены основные особенности и возможности средств имитационного моделирования цифровых систем на основе HDL-языков. Рассмотрен пример задачи, решаемой студентами в процессе изучения архитектуры вычислительных систем с использованием средств языка Verilog и основанных на нем технологий. В качестве примера использована модель интерфейсного элемента для передачи данных по последовательному каналу связи. Приведен вариант решения задачи, включающий описание изучаемого компонента, технологии тестирования его логической модели с использованием пакета Icarus Verilog, а также варианты наглядного представления результатов тестирования логической модели интерфейса средствами программы GTKWave. Приведен примерный список вопросов и заданий для контроля уровня усвоения учебного материала.

Alexeevskiy Petr Ivanovich,
Assistant Lecturer of Department of Informatics, Computer Technology and Methods of Teaching Informatics, Ural State Pedagogical University, Ekaterinburg, Russia.

TRAINING OF STUDENTS IN THE ARCHITECTURE OF COMPUTER SYSTEMS USING THE VERILOG LANGUAGE

KEYWORDS: computer systems architecture; computer systems design; computation devices; hardware definition languages.

ABSTRACT. In this article the need for the study of the processor core and basic peripheral devices in the process of training students in the architecture of computer systems is defined, considering requirements of the education standards and technologies used in production. The possibilities and prospects for the use of the Verilog HDL in the process of students training in the architecture of computer are analyzed. Main features of the Verilog language, which are important for educational purposes, are listed. The main features and opportunities of HDL-based imitational modeling of digital systems are explored. An example a task for the students to solve during the study of computer systems architecture based on the Verilog HDL and related technologies is also described. As an example, a model of the serial line interface unit is used. A solution is provided, which include a description of the component, a testbench for its logic model testing using Icarus Verilog, as well as the test results presentation cases using GTKWave. A list of test questions and tasks to measure the mastery level of education materials is provided.

Современные темпы и особенности развития вычислительной техники позволяют повысить эффективность программно-аппаратных решений в различных областях науки и техники. В то же время разработка решений на основе обычных микропроцессорных систем общего назначения ограничивается возможностями используемых микропроцессоров и подходами к их использованию. Для решения задач, где возможностей микропроцессоров (МП) и микроконтроллеров (МК) недостаточно, обычно применяются предметно-специфичные устройства (ASIC, Application-Specific Integrated Circuit). Такие устройства обычно либо работают в совокупности с МП или МК общего назначения, либо включают собственное процессорное ядро, повышая тем самым гибкость и масштабируемость программно-аппаратного решения в целом.

Изучение архитектуры вычислительных систем (ВС) в вузах, как правило, ограничивается лишь описанием работы конкретных процессорных архитектур (таких как IA-32, AMD64, ARM и др.), а также особенностей низкоуровневой реализации простых алгоритмов на этих архитектурах. Решение каких-либо сложных задач обработки данных при этом предполагается исключительно программным способом, что зачастую не является оптимальным. Изучению возможностей применения ASIC обыч-

© Алексеевский П. И., 2016

но не уделяется внимания в рамках курса архитектуры ВС; кроме того, изучение работы МП или МК не подразумевает изучения устройства их отдельных элементов (исполнительных блоков, декодера инструкций, шины управления и т. п.).

Необходимость изучения принципов построения ASIC, МП и МК обусловлена также требованиями ФГОС ВО по различным направлениям (02.03.02 — ОПК-2, ОПК-3 [3]; 09.03.02 — ОПК-6, ПК-5, ПК-12 [4]; 44.03.01 — ПрК-8, ПрК-9 [5] и др.).

Изучение устройства МП, МК и других вычислительных узлов в рамках курса архитектуры ВС в вузах может быть организовано путем моделирования компонентов процессорного ядра программными [4] и/или аппаратными средствами. В качестве программного средства имитационного моделирования при создании сложных вычислительных устройств обычно используются логические симуляторы, входящие в состав различных пакетов EDA (таких как Altera Quartus, Xilinx ISE и др.) либо поставляемых независимо (Icarus Verilog, Active-HDL и др.); для тестирования аппаратной реализации применяются отладочные наборы на базе CPLD (Complex Programmable Logic Device, сложное программируемое логическое устройство) или FPGA (Field-Programmable Gate Array, программируемый полем массив логических вентилей). Для того чтобы сформировать у учащихся представление о работе таких устройств, требуется по крайней мере тестирование модели соответствующего компонента с помощью программного симулятора.

Разработка вычислительных устройств для последующей их реализации в форме ASIC осуществляется средствами специализированных языков описания оборудования (HDL, Hardware Description Language), включающих VHDL, Verilog, G, AHDL, CUPL, ABEL и др. Наибольшее распространение среди них имеют языки Verilog и VHDL.

Несмотря на сходство HDL с языками программирования общего назначения, существует ряд особенностей, обусловленных спецификой предмета. Например, последовательность операторов таких языков не гарантирует последовательное их исполнение — в большинстве случаев эти операторы выполняются параллельно. Хотя последовательное исполнение операторов возможно, такое поведение модели не является основным.

Выбор программной платформы, применяемой для обучения устройству МП и МК, в случае использования отладочных комплектов обусловлен имеющимся в наличии оборудованием и составом тестового стенда. В общем же случае, если требуется только имитационное моделирование, основополагающим фактором являются характеристики используемых ПК.

В таблице 1 приведены особенности различных платформ, позволяющих изучать устройство компонентов МП/МК путем имитационного моделирования.

Таблица 1

Программные платформы, реализующие имитационное моделирование

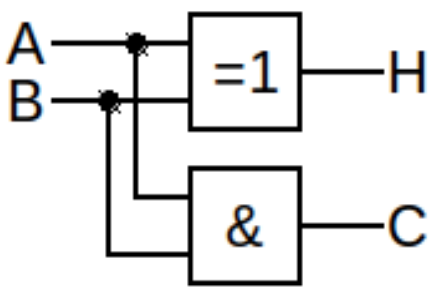
Платформа	**Разработчик**	**Лицензия**	**Поддерживаемые ОС**			**Особенности**
			Windows	**GNU/Linux**	**Другие**	
Altera Quartus	Altera/Intel	Проприетарная	+	+	-	Синтез для CPLD и FPGA от Altera — MAX, Cyclone, Arria и др. [14]
Xilinx ISE	Xilinx	Проприетарная	+	+	-	Синтез для CPLD и FPGA от Xilinx — Spartan, Virtex, Zynq и др. [11]
LabVIEW	National Instruments	Проприетарная	+	+	+	Общего назначения, только язык G. [12]
Active-HDL	Aldec	Проприетарная	+	-	-	Общего назначения [7]
Icarus Verilog	Stephen Williams	GPL-2	+	+	+	Общего назначения, только симуляция [15]

По совокупности рассмотренных параметров (универсальность, доступность, возможность работы под управлением различных ОС) для рассмотрения в процессе реализации дисциплин «Архитектура вычислительных систем» и «Архитектура компьютера» в Институте математики, информатики и информационных технологий УрГПУ была выбрана платформа Icarus Verilog.

Платформа Icarus Verilog включает два основных компонента – транслятор с языка Verilog (iverilog) и симулятор (vvp). Оба этих компонента являются консольными программами, управление которыми осуществляется преимущественно средствами аргументов командной строки, что позволяет встраивать их в автоматизированный процесс построения и тестирования сложного программно-аппаратного проекта.

Язык Verilog имеет простой синтаксис, что позволяет изучить особенности его синтаксиса и приступить к практической реализации учебных моделей вычислительных устройств в сжатые сроки. Средства языка поддерживают как описание непосредственно логической модели, так и средства тестирования для использования в совокупности с симулятором [6; 7].

Изучение принципов работы вычислительных устройств следует начинать с простейших элементов, таких как простые комбинационные схемы, триггеры, защелки и т.п. На рисунке 1 приведен пример схемы полусумматора и соответствующая ему Verilog-модель.




```
module half_adder(A, B, H, C);
input A, B;
output H, C;
assign H = A ^ B;
assign C = A & B;
endmodule
```

Рис. 1
Схема и Verilog-модель полусумматора

Поскольку простейшие элементы в рамках некоторого сложного вычислительного устройства многократно повторяются, следует акцентировать внимание студентов на возможности повторного использования реализованных модулей. Практически все современные трансляторы языка Verilog позволяют объединить множество модулей в единую модель.

Синтаксис языка Verilog включает директивы, предназначенные для управления процессом обработки модели в симуляторе [10]. Они позволяют наблюдать за изменением сигналов с привязкой ко времени, выводить отладочные сообщения при работе узлов, приостанавливать выполнение, использовать данные из внешних файлов для тестирования, а также осуществлять интерактивное управление моделью. Эти возможности позволяют студентам изучить представленные в наглядной форме результаты выполнения моделью заданных операций, отслеживать ошибки и т.д. Таким образом, приведенная модель полусумматора может быть использована в тестовом модуле, приведенном на рисунке 2.

```
module ha_test;
initial begin
  #10 {A, B} <= 2'b00;
  #10 {A, B} <= 2'b01;
  #20 {A, B} <= 2'b10;
  #20 {A, B} <= 2'b11;
  #50 $stop;
end
reg A, B;
wire H, C;
half_adder ha1(A, B, H, C);
initial
  $monitor("T=%t A=%b B=%b H=%b C=%b",
        $time, A, B, H, C);
endmodule
```

а)

```
T=                   0 A=x B=x H=x C=x
T=                  10 A=0 B=0 H=0 C=0
T=                  20 A=0 B=1 H=1 C=0
T=                  40 A=1 B=0 H=1 C=0
T=                  60 A=1 B=1 H=0 C=1
** VVP Stop(0) **
** Flushing output streams.
** Current simulation time is 110 ticks.
```

б)

Рис. 2
Тестовая модель (а) и результат ее обработки симулятором (б).

После изучения синтаксиса языка Verilog и основных приемов работы с симулятором студенты могут приступить к изучению более сложных компонентов:

– мультиплексоры и демультиплексоры;

– исполнительные блоки процессора (многоразрядные сумматоры, блоки выполнения логических операций);

– блоки управления шиной адреса и шиной данных;

– декодер инструкций процессора;

– регистр состояния процессора;

– конвейеры инструкций.

Ввиду большого количества различных сложных блоков, перед студентами не ставится задача реализации модели какого-либо сложного МП или МК. Несмотря на развитие элементной базы, алгоритмов обработки данных, и как следствие — существенный рост производительности процессоров, в основе современных МП и МК по прежнему лежат подходы, сформулированные в 1930–1950 гг. Таким образом, в качестве учебного практического материала может быть использовано устройство простейшего процессора, поддерживающего минимальный набор операций.

После изучения внутреннего устройства процессорного ядра имеет смысл перейти к изучению периферийных устройств. На этом этапе можно взять за основу модель более сложного процессора, в том числе основанного на какой-либо из используемых на практике архитектур (таких как x86, ARM или MIPS). Готовые к использованию функциональные модели таких процессоров находятся в свободном доступе в Интернете.

Использование модели современного процессора позволяет студентам осуществлять отладку собственных моделей периферийных устройств, используя для разработки программной части проекта соответствующее инструментальное и системное ПО (ассемблеры, компиляторы языков программирования высокого уровня и в некоторых случаях — операционные системы).

Основные периферийные устройства, модель которых имеет смысл строить в рамках курса архитектуры ВС, включают в себя:

1) программируемый интервальный таймер (Intel i8253, i8254 или аналогичный);

2) программируемый контроллер прерываний (Intel i8259, AMD Am9519 и т. п.);

3) контроллер прямого доступа к памяти (Intel i8257).

Общие приемы работы с большинством из приведенных периферийных устройств рассматривается во всех реализуемых в настоящее время курсах архитектуры ВС, поскольку данные устройства в упрощенном или расширенном виде присутствуют во всех современных IBM PC-совместимых материнских платах ПК в составе ASIC системной логики с целью обеспечения совместимости. Однако изучение принципов, лежащих в основе этих устройств, позволяет не только наиболее полно рассмотреть их возможности, но и дать возможность студентам оптимизировать их работу и расширить функциональность. Для этого можно сформулировать следующие дополнительные задания для лабораторных работ:

1. Расширение функционала программируемого интервального таймера i8253 путем добавления новых видов выходного сигнала (дополнительно к меандру и коротким импульсам) – управление скважностью выходного сигнала для использования таймера в качестве ШИМ-контроллера.

2. Расширение функционала программируемого контроллера прерываний i8259 – упрощение настройки приоритетов прерываний; добавление режимов работы, совместимых с МП, отличными от i8080/8085/8086 (например, Zilog Z80 и его расширенные версии).

3. Расширение функционала контроллера прямого доступа к памяти i8257 – упрощение работы контроллера с шиной; расширение шины адреса для возможности использования регионов памяти размером больше 64кБ.

В качестве примера использования языка Verilog рассмотрим задачу реализации модели интерфейса передачи данных по последовательному каналу связи, аналогичному RS-232 [8]. При выполнении задания студенты должны:

1) проанализировать механизм передачи данных по последовательному каналу связи;

2) определить состав передающего устройства и способы управления процессом передачи;

3) построить Verilog-модель интерфейса;

4) создать средство тестирования построенной модели;

5) осуществить имитационное моделирование работы передающего устройства с помощью симулятора;

6) представить результаты тестирования в наглядной форме и ответить на вопросы.

Данная задача предполагает, что информация по последовательному каналу связи передается кадрами длиной по 10 бит (1 стартовый бит + передаваемый байт + 1 стоповый бит). Передача битов данных осуществляется, начиная со старшего разряда. Поскольку загружать в передатчик новое значение во время передачи данных нельзя, требуется обеспечить формирование сигнала занятости линии. Таким образом, передатчик должен иметь, по крайней мере, три входа (восьмиразрядная шина данных, вход команды загрузки

значения и вход тактовых импульсов) и два выхода (линия передачи данных и признак активности).

Для хранения передаваемого кадра целесообразно использовать десятиразрядный сдвиговый регистр (по размеру кадра). По переднему фронту тактового сигнала содержимое регистра должно сдвигаться влево, вытесняя уже переданный старший разряд. Освободившийся младший разряд заполняется значением логической 1 (соответствующий уровень присутствует на линии при отсутствии активности). Для подсчета количества сдвигов и формирования сигнала занятости имеет смысл использовать двоичный счетчик – в таком случае признак занятости линии вычисляется как результат логического ИЛИ всех битов текущего значения счетчика.

После анализа механизма передачи данных студенты могут реализовать соответствующую Verilog-модель. Пример реализации модели последовательного передатчика приведен на рисунке 3.

```
module sif(input [7:0]data, input clock, input load,
           output line, output hold);
reg [9:0]buffer;
reg [3:0]counter;
assign hold = |counter;
assign line = buffer[9];
initial buffer = 10'b1_1111_1111_1;
initial counter = 4'd0;
always @(posedge clock) begin
  if (load) begin
    buffer <= {1'b0, data, 1'b1};
    counter <= 4'd10;
  end else begin
    if (counter > 0) begin
      counter <= counter - 1;
      buffer[9:0] = {buffer[8:0], 1'b1};
    end
  end
end
endmodule
```

Рис. 3
Verilog-модель интерфейса последовательной передачи данных

После реализации модели ее необходимо протестировать. Для тестирования модели студентам потребуется создать еще один модуль, в котором будет создаваться экземпляр данного передатчика и соответствующий набор переменных для управления процессом передачи. Поскольку передача данных опирается на тактовый сигнал, необходимо создать источник такого сигнала – для этого достаточно через равные интервалы времени инвертировать значение однобитной переменной.

Если передатчик реализован в соответствии с рисунком 3, то загрузка очередного байта в сдвиговый регистр осуществляется по переднему фронту сигнала load. Таким образом, для загрузки байта необходимо установить его на шине данных и кратковременно (по крайней мере, на половину периода основного тактового сигнала) перевести линию load в состояние логической 1. Передача значения в этом случае начинается автоматически, и перед отправкой следующего байта необходимо дождаться освобождения линии (задний фронт сигнала hold).

Для получения информации о процессе тестирования модели студентам следует предварительно выбрать формат, в котором эта информация будет сохраняться. Если требуется получать предварительно обработанные данные – в реализации модели для вывода данных могут быть использованы конструкции языка Verilog, такие как $monitor, $display и т. п. [13]. Это позволяет получить информацию в формате CSV, либо протокол работы произвольного формата. Если же требуется получить информацию о состоянии всех линий в каждый момент времени, то целесообразно использовать автоматический сбор соответствующих данных в файл специального формата, который может быть просмотрен и обработан соответствующими программами (например, GTKWave [9]).

Пример реализации тестового модуля приведен на рисунке 4. Данная реализация осуществляет вывод информации как в формате CSV, так и в специальном формате, поддерживаемом программой GTKWave.

```
module sif_test;
reg [7:0]txdata;
reg clock, load;
wire line, hold;
initial forever #5 clock = ~clock;
initial begin
  $dumpfile("test.vcd");
  $dumpvars(0, sif_test);
  $display("Time,Line,Hold");
  $monitor("%0t,%b,%b", $time, line, hold);
  load = 0;
  clock = 0;
                  #100 txdata = "T"; load = 1; #6 load = 0;
  @(negedge hold) #100 txdata = "e"; load = 1; #6 load = 0;
  @(negedge hold) #100 txdata = "s"; load = 1; #6 load = 0;
  @(negedge hold) #100 txdata = "t"; load = 1; #6 load = 0;
  #1000 $finish;
end
sif sif1(txdata, clock, load, line, hold);
endmodule
```

Рис. 4
Тестовый модуль

После обработки данных модулей транслятором, полученный файл может быть обработан симулятором (при использовании Icarus Verilog для этого используется программа vvp). Консольный вывод симулятора может быть захвачен в CSV-файл для дальнейшей обработки в виде электронной таблицы. Состояние всех сигналов в приведенном примере будет сохранено в файл test.vcd. Для представления этого состояния в наглядной форме учащиеся могут загрузить полученный файл в программу GTKWave. Результат для приведенного примера изображен на рисунке 5.

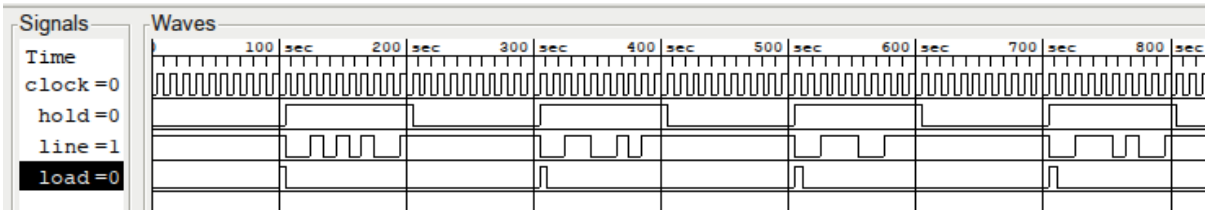


Рис. 5
Результаты обработки модели в программе GTKWave

На рисунке видно, что передача данных по линии связи (строка line) осуществляется четырьмя кадрами. Передаваемая по линии информация (строка «Test», см. рисунок 4) загружается в сдвиговый регистр по переднему фронту сигнала load, при этом непосредственно передача начинается по следующему переднему фронту тактового сигнала (строка clock). Во время передачи данных сигнал hold удерживается на уровне логической 1, что позволяет оценить на графике границы и длину передаваемого кадра.

После выполнения приведенного задания студентам могут быть заданы следующие вопросы:

1. Требуются ли какие-либо дополнительные сигналы, чтобы передаваемые данные могли быть корректно распознаны принимающим устройством?

2. Какое влияние на передачу данных окажет различие фаз тактовых сигналов принимающего и передающего устройств?

3. Что произойдет, если периоды тактовых сигналов принимающего и передающего устройств будут различаться на величину большую, чем время передачи одного бита? На меньшую величину?

4. Какое влияние на процесс передачи данных окажет загрузка в сдвиговый регистр нового значения, пока линия hold удерживается на уровне логической 1? Возможно ли восстановление нормальной работы канала после возникновения такой ситуации?

5. Что произойдет, если удерживать сигнал load на уровне логической 1 дольше одного периода тактового сигнала?

Анализируя правильность ответов на поставленные вопросы, определяется уро-

вень усвоения рассмотренной темы. Для закрепления материала могут быть заданы дополнительные задания:

1) расчет и добавление кода четности/нечетности в передаваемый кадр;

2) реализация помехоустойчивого кодирования передаваемых данных;

3) реализация модели принимающего устройства.

Таким образом, изучение языков HDL, реализация функциональных моделей элементов центрального процессора и простых периферийных устройств, а также осуществление их имитационного моделирования средствами логического симулятора позволит формировать у студентов знания и умения в полном соответствии с требованиями ФГОС ВО. Использование языков HDL для синтеза предметно-специфичных устройств на основе CPLD или FPGA дает возможность решать задачи, выходящие за пределы возможностей МП и МК общего назначения. Полученные знания позволят выпускникам создавать программно-аппаратные комплексы для наиболее оптимального и эффективного решения производственных задач.

ЛИТЕРАТУРА

1. Авдеев В. А. Компьютерное моделирование цифровых устройств. М. : ДМК Пресс, 2012. 366 с.

2. Головков А. А., Пивоваров И. Ю., Кузнецов И. Р. Компьютерное моделирование и проектирование радиоэлектронных средств : учебник для вузов. Стандарт третьего поколения. СПб. : Питер. 2015. 208 с.

3. Приказ Минобрнауки России от 04.12.2015 N 1426 «Об утверждении федерального государственного образовательного стандарта высшего образования по направлению подготовки 44.03.01 Педагогическое образование (уровень бакалавриата)». URL: http://www.consultant.ru/document/cons_doc_LAW_192459/ (дата обращения 18.04.2016).

4. Приказ Минобрнауки России от 12 марта 2015 года № 219 «Об утверждении федерального государственного образовательного стандарта высшего образования по направлению подготовки 09.03.02 информационные системы и технологии (уровень бакалавриата)». URL: http://минобрнауки.рф/документы/5433 (дата обращения 18.04.2016).

5. Приказ Минобрнауки России от 12 марта 2015 года № 224 «Об утверждении федерального государственного образовательного стандарта высшего образования по направлению подготовки 02.03.02 Фундаментальная информатика и информационные технологии (уровень бакалавриата)». URL: http://минобрнауки.рф/документы/5431 (дата обращения 18.04.2016).

6. Соловьев В. В. Основы языка проектирования цифровой аппаратуры Verilog. М. : Горячая линия-Телеком, 2016. 208 с.

7. Active-HDL — FPGA Simulation // Aldec. URL: https://www.aldec.com/en/products/fpga_simulation/active-hdl (дата обращения 19.04.2016).

8. EIA standard RS-232-C: Interface between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange. // Washington: Electronic Industries Association. Engineering Dept. 1969. 29 p.

9. GTKWave 3.3 Wave Analyzer User's Guide. BSI. 2016. URL: http://gtkwave.sourceforge.net/gtkwave.pdf (дата обращения 16.05.2016).

10. IEEE Std 1364-2001 / IEEE Standard Verilog Hardware Description Language. // IEEE. 2001. URL: https://inst.eecs.berkeley.edu/~cs150/fa06/Labs/verilog-ieee.pdf (дата обращения 19.04.2016).

11. ISE Design Suite. Xilinx. URL: http://www.xilinx.com/products/design-tools/ise-design-suite.html (дата обращения 19.04.2016).

12. LabVIEW FPGA Module // National Instruments. URL: http://www.ni.com/labview/fpga/ (дата обращения 19.04.2016).

13. Palnitkar S. Verilog HDL: A Guide to Digital Design and Synthesis, Second Edition. Prentice Hall PTR. 2003. 450 p.

14. Quartus Prime — Overview // Altera. URL: https://www.altera.com/products/design-software/fpga-design/quartus-prime/overview.html (дата обращения 19.04.2016).

15. Williams S. Icarus Verilog — Simulation / Wikia. URL: http://iverilog.wikia.com/wiki/Simulation (дата обращения 19.04.2016).

LITERATURA

1. Avdeev V. A. Komp'yuternoe modelirovanie tsifrovykh ustroystv. M. : DMK Press, 2012. 366 s.

2. Golovkov A. A., Pivovarov I. Yu., Kuznetsov I. R. Komp'yuternoe modelirovanie i proektirovanie radioelektronnykh sredstv : uchebnik dlya vuzov. Standart tret'ego pokoleniya. SPb. : Piter. 2015. 208 s.

3. Prikaz Minobrnauki Rossii ot 04.12.2015 N 1426 «Ob utverzhdenii federal'nogo gosudarstvennogo obrazovatel'nogo standarta vysshego obrazovaniya po napravleniyu podgotovki 44.03.01 Pedagogicheskoe obrazovanie (uroven' bakalavriata)». URL: http://www.consultant.ru/document/cons_doc_LAW_192459/ (data obrashcheniya 18.04.2016).

4. Prikaz Minobrnauki Rossii ot 12 marta 2015 goda № 219 «Ob utverzhdenii federal'nogo gosudarstvennogo obrazovatel'nogo standarta vysshego obrazovaniya po napravleniyu podgotovki 09.03.02 informatsionnye sistemy i tekhnologii (uroven' bakalavriata)». URL: http://minobrnauki.rf/dokumenty/5433 (data obrashcheniya 18.04.2016).

5. Prikaz Minobrnauki Rossii ot 12 marta 2015 goda № 224 «Ob utverzhdenii federal'nogo gosudarstvennogo obrazovatel'nogo standarta vysshego obrazovaniya po napravleniyu podgotovki 02.03.02 Fundamental'naya informatika i informatsionnye tekhnologii (uroven' bakalavriata)». URL: http://minobrnauki.rf/dokumenty/5431 (data obrashcheniya 18.04.2016).

6. Solov'ev V. V. Osnovy yazyka proektirovaniya tsifrovoy apparatury Verilog. M. : Goryachaya liniya-Telekom, 2016. 208 s.

7. Active-HDL — FPGA Simulation // Aldec. URL: https://www.aldec.com/en/products/fpga_simulation/active-hdl (data obrashcheniya 19.04.2016).

8. EIA standard RS-232-C: Interface between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange. // Washington: Electronic Industries Association. Engineering Dept. 1969. 29 p.

9. GTKWave 3.3 Wave Analyzer User's Guide. BSI. 2016. URL: http://gtkwave.sourceforge.net/gtkwave.pdf (data obrashcheniya 16.05.2016).

10. IEEE Std 1364-2001 / IEEE Standard Verilog Hardware Description Language. // IEEE. 2001. URL: https://inst.eecs.berkeley.edu/~cs150/fa06/Labs/verilog-ieee.pdf (data obrashcheniya 19.04.2016).

11. ISE Design Suite. Xilinx. URL: http://www.xilinx.com/products/design-tools/ise-design-suite.html (data obrashcheniya 19.04.2016).

12. LabVIEW FPGA Module // National Instruments. URL: http://www.ni.com/labview/fpga/ (data obrashcheniya 19.04.2016).

13. Palnitkar S. Verilog HDL: A Guide to Digital Design and Synthesis, Second Edition. Prentice Hall PTR. 2003. 450 p.

14. Quartus Prime — Overview // Altera. URL: https://www.altera.com/products/design-software/fpga-design/quartus-prime/overview.html (data obrashcheniya 19.04.2016).

15. Williams S. Icarus Verilog — Simulation / Wikia. URL: http://iverilog.wikia.com/wiki/Simulation (data obrashcheniya 19.04.2016).

Статью рекомендует д-р пед. наук, проф. Б. Е. Стариченко