ОБУЧЕНИЕ ПО АВТОМАТИЗИРАНО ПРОЕКТИРАНЕ НА ЦИФРОВИ СХЕМИ

Петър Борисов Минев, Валентиина Стоянова Кукенска[1], Матьо Стефанов Динев[1]
[1]Технически университет - Габрово

TEACHING FOR AUTOMATION DESIGN OF DIGITAL CIRCUITS

Petar Borisov Minev, Valentina Stoyanova Kukenska[1], Matyo Stefanov Dinev[1]
[1] Technical University of Gabrovo

Abstract

This report presents a methodology for practical training of students in disciplines related to automated design of digital circuits and systems from the curriculum of department "Computer Systems and Technologies". The methodology is based on the principle of hierarchical design of digital hardware designs with VHDL language. This project is realized on school boards with FPGA, using the accompanying software development environments. The methodology is demonstrated by creating a sample project of a digital circuit with VHDL.

Keywords: Teaching; VHDL programming; FPGA; Automation design tools; Digital circuits.

ВЪВЕДЕНИЕ

Автоматизираното проектиране на цифрови схеми се изучава в група от избираеми и задължителни дисциплини от бакалавърската и магистърска програми на специалност "Компютърни системи и технологии" в Технически университет – Габрово.

Студентите получават знания по основните методи за функционално, логическо и структурно проектиране на цифрови схеми и системи, включително езици за хардуерно описание на електронни схеми, като VHDL, особености при проектирането с програмируеми логически интегрални схеми (FPGA) и тяхното приложение, както и умения за работа със системи за автоматизирано проектиране на цифров хардуер.

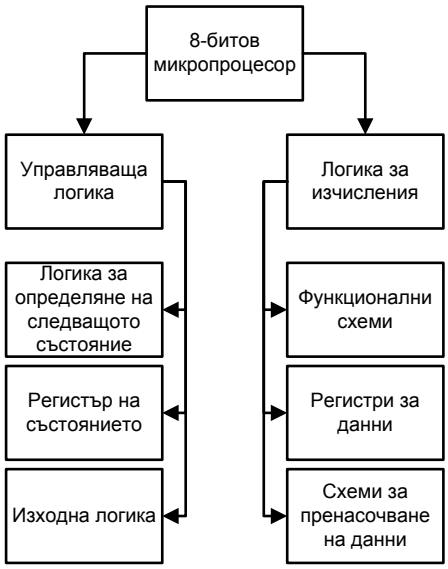
На студентите са необходими предварителни знания по следните дисциплини: Анализ и синтез на логически схеми, Цифрова схемотехника, Автоматизация на инженерния труд, Микропроцесорна техника, Компютърна периферия.

За подпомагане на обучението е създадена библиотека от модели на стандартни цифрови схеми, като броячи, шифратори и дешифратори, мултиплексори, различни видове регистри, суматори, умножители, компаратори и др. Тя се използва от студентите при разработването на йерархични проекти в лабораторните упражнения. Така в процеса на тяхното обучение се прилага един от основните принципи при проектиране на цифрови схеми – йерархичното проектиране.

Йерархичното проектиране е методология, при която една схема се разделя рекурсивно на по-малки модули, след което всеки модул се проектира независимо от останалите. При необходимост процесът на разделяне може да продължи с декомпозиране на вече разделените модули. Така, една сложна схема последователно се разделя на подсхеми, позволявайки на студента да се фокусира върху една малка и лесна за проектиране част от схемата. [1]

Рекурсивното разделяне на една схема на модули и последващото декомпозиране на вече разделените модули, позволяващо проектиране на всеки модул независимо от останалите е представено на фиг. 1. Микропроцесорът е разделен на логика за управление и логика за изчисления [2]. Логиката за управление се подразделя на: логика за определяне на следващото състояние; регистър на състоянието и изходна логика. Логиката за изчисления се разделя на: схеми за пренасочване на данните; функционални схеми и регистри за данни. Функционалните схеми се представят с аритметико-логическо устройство, регистрите за данни

– с регистров файл, а схемите за пренасочване на данните се заменят с мултиплексори. В управляващата логика един програмен брояч заменя регистъра на състоянието, а изходната логика се заменя с памет за инструкции и декодираща инструкциите логика. Ако е необходимо процесът на декомпозиция може да продължи, за да се намали още повече сложността на модулите. Например, АЛУ може да се раздели на суматор, умножител, схема за сравнение и пр.

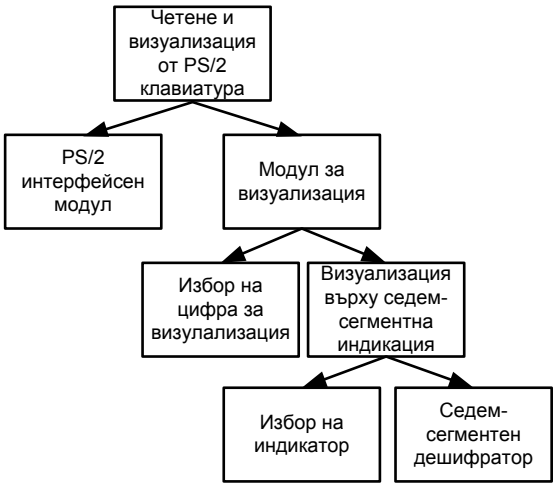


***Фиг. 1**. Йерархично описание на 8-битов микропроцесор*

ИЗЛОЖЕНИЕ

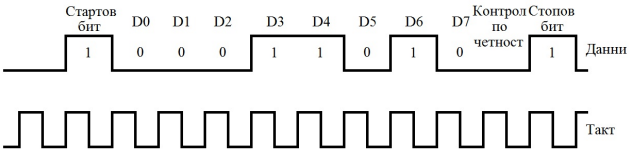
Използването на принципа на йерархичната декомпозиция в процеса на обучение по автоматизирано проектиране на цифрови схеми е демонстрирано чрез представянето на учебен проект на схема за приемане на данни за натиснат клавиш от клавиатура и визуализацията им върху седемсегментна индикация.

Разделянето на проекта на модули е представено на фиг. 2. Първоначално схемата е разделена на интерфейсен модул, който да приема постъпващите данни по PS/2 интерфейса на клавиатурата и модул за визуализация на приетите данни върху седемсегментна индикация. Модулът за визуализация от своя страна може да бъде разделен на две подсхеми. Една за избор на цифра, която да се визуализира и друга за преобразуване на цифрата в код за визуализация върху седемсегментен индикатор. Обикновенно седемсегментната индикация се състои от два, четири или повече индикатора. Това налага модулът „Визуализация върху седемсегментна индикация" да се раздели на схема за избор на индикатор и седемсегментен дешифратор, преобразуващ двоичния код на цифрата в седемсегментен код.



***Фиг. 2**. Йерархично описание на схема за четене и визуализацията на данни за натиснат клавиш*

Интерфейсът на клавиатурата се състои от две комуникационни линии: линия за тактов сигнал и линия за данни. Един байт с данни се изпраща в единайсет битов пакет, който съдържа: стартов бит; осем информационни бита; бит за проверка по четност и стопов бит. Формата за предаване на данните и съответстващите им тактови импулси са показани на фиг. 3.



***Фиг. 3**. Времедиаграма на преноса на PS2 данни*

PS/2 интерфейсният модул осигурява връзката между клавиатурата и останалата част от проектираната схема. Той поддържа комуникационния протокол с клавиатурата и осигурява преобразуването на последователния код в паралелен [3]. За неговата реа-

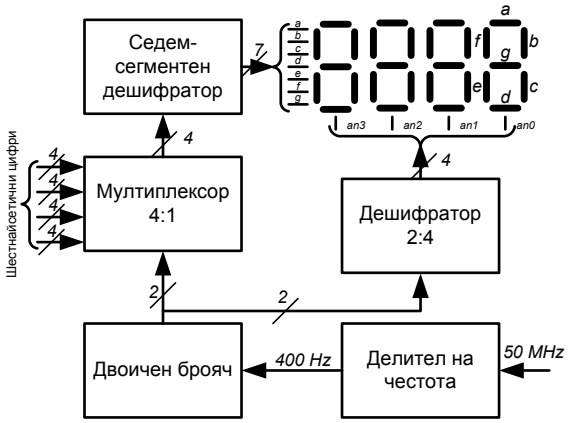
лизация може да се използва преместващ регистър. Част от програмния код на VHDL на преместващ регистър за четене от PS/2 клавиатура е представен на фиг. 4.

```
5
6   ENTITY SR IS
7     PORT ( rst : in  STD_LOGIC;
8          PS2c : in  STD_LOGIC;
9          PS2d : in  STD_LOGIC;
10         keycode : out  STD_LOGIC_VECTOR(7 downto 0));
11  END SR;
```

Фиг. 4. *VHDL описание на PS/2 инетрфейсен модул за клавиатура*

Модулът за визуализация на приетия от клавиатурата байт с данни може да се реализира като стандартна схема за динамична седемсегментна индикация. Модулите, от които се изгражда динамична седемсегментна индикация са: седемсегментен дешифратор; схема за избор на цифра, която да се визуализира върху индикаторите в даден момент; схема за избор на индикатор, който да бъде включен в същия момент, в който е избрана и съответната цифра; схема, осигуряваща необходимата честота за синхронно превключване на индикаторите и цифрите.

Схемата на динамична седемсегментна четири разрядна индикация е представена в блоков вид на фиг. 5.



Фиг. 5*. Схема на четириразрядна 7-сегмантна идникация*

Мултиплексорът в схемата от фиг. 5 превключва между подадените на входовете му цифри, така че на входа на седемсегментния дешифратор в даден момент е подадена само една от тях. Дешифтаторът преобразува двоичния код на подадената цифра в позиционен седемсегментен код, показващ кои светодиоди от индикатора ще светят. Превключването между индикаторите се извършва от стандартен дешифтарор, който е синхронизиран с превключването на цифрите от мултиплексора. За тази синхронозация се използва изхода на двубитов брояч, който генерира последователно двоичните стойности на числата от нула до две. Скоростта с която работи броячът определя и скоростта на превключване. За получаване на образ без трептене е удачно да се използва тактова честота от няколко стотин херца. Такава честота може да се получи чрез делител на честота, който да намали честотата на тактовия сигнал от вградения кварцов генаратор върху учебните развойни платки с програмируема логика, използвани в лабораторните упражнения. Делителят на честота може да се реализира чрез многоразреден брояч, както е показано в [4].

Интерфейсните части на програмните кодове на VHDL за представените модули, изграждащи схемата на описаната динамична индикация са представени на фигури 6, 7, 8, 9 и 10.

```
32  entity mux is
33    Port ( sel : in  STD_LOGIC;
34         dig_0 : in  STD_LOGIC_VECTOR (3 downto 0);
35         dig_1 : in  STD_LOGIC_VECTOR (3 downto 0);
36         dig_2 : in  STD_LOGIC_VECTOR (3 downto 0);
37         dig_3 : in  STD_LOGIC_VECTOR (3 downto 0);
38         hex_dig : out  STD_LOGIC_VECTOR (3 downto 0));
39  end mux;
40
```

Фиг. 6*. VHDL описание на интерфейс на мултиплексор*

```
23  entity seven_seg_dec is
24    Port ( hex_dig : in  STD_LOGIC_VECTOR (3 downto 0);
25         ssd_an : out  STD_LOGIC_VECTOR (6 downto 0));
26  end seven_seg_dec;
```

Фиг. 7*. VHDL описание на интерфейс на 7-сегментен дешифратор*

```
32  entity dec2to1 is
33    Port ( a : in  STD_LOGIC_VECTOR (1 downto 0);
34         b : out  STD_LOGIC_VECTOR (3 downto 0));
35  end dec2to1;
```

Фиг. 8*. VHDL описание на интерфейс на стандартен дешифратор 2:4*

```
33  entity counter2 is
34    Port ( clk : in  STD_LOGIC;
35         rst : in  STD_LOGIC;
36         count : out  STD_LOGIC_VECTOR (1 downto 0));
37  end counter2;
```

Фиг. 9*. VHDL описание на интерфейс на двоичен брояч*

```
33  entity reg_div is
34      Port ( mclk : in  STD_LOGIC;
35             rst : in  STD_LOGIC;
36             clk_div : out  STD_LOGIC);
37  end reg_div;
```

***Фиг. 10**. VHDL описание на интерфейс на делител на честота*

За изграждане на йерархичния проект във VHDL, свързващ представените VHDL модули се използва шаблон, който е показан на фиг. 11.

Различните модули, участващи в йерархичния проект се декларират като компоненти с техните имена и портове. За свързване между компоненти се декларират сигнали, като за всяка връзка се използва отделен сигнал. Декларираните компоненти се включват във VHDL архитектурата, като се указва какво е свързването между портовете на включения компонент, портовете на общата структура и вътрешните сигналите за свързване.

```
32  entity compreg is
33      Port ( a : in  STD_LOGIC_VECTOR (7 downto 0);
34             ena : in  STD_LOGIC;
35             rst : in  STD_LOGIC;
36             clk : in  STD_LOGIC;
37             b : in  STD_LOGIC_VECTOR (7 downto 0);
38             eq : out  STD_LOGIC;
39             lt : out  STD_LOGIC;
40             gt : out  STD_LOGIC);
41  end compreg;
42
43  architecture Behavioral of compreg is
44
45  component reg
46  port (d : in  STD_LOGIC_VECTOR (7 downto 0);
47             clk : in  STD_LOGIC;
48             rst : in  STD_LOGIC;
49             ena : in  STD_LOGIC;
50             q : out  STD_LOGIC_VECTOR (7 downto 0));
51             end component;
52
53     component comparator1
54     Port ( a : in  STD_LOGIC_VECTOR (7 downto 0);
55             b : in  STD_LOGIC_VECTOR (7 downto 0);
56             eq : out  STD_LOGIC;
57             lt : out  STD_LOGIC;
58             gt : out  STD_LOGIC);
59             end component;
60
61     signal t1,t2: STD_LOGIC_VECTOR (7 downto 0);
62
63  begin
64  u1: reg port map( d=>a, ena=>ena, rst=>rst, clk=>clk, q=>t1);
65  u2: reg port map( d=>b, ena=>ena, rst=>rst, clk=>clk, q=>t2);
66  u3: comparator1 port map( a=>t1, b=>t2, eq=>eq, lt=>lt, gt=>gt);
67  end Behavioral;
```

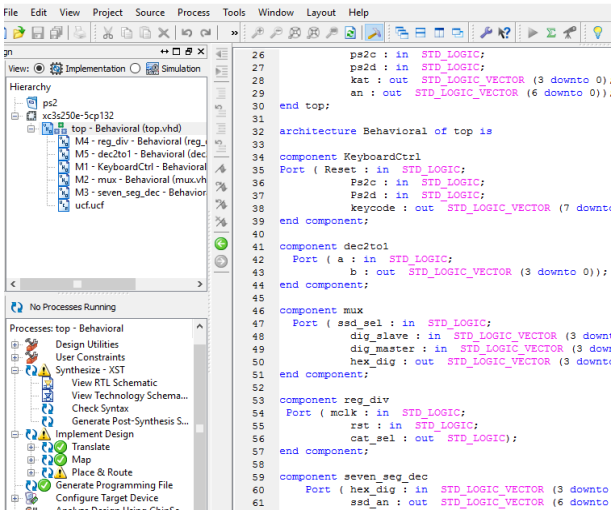
Декларирване на компонент

Деклариране на сигнал

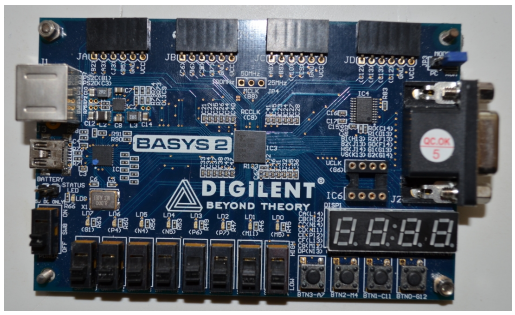
Включване на компонент

***Фиг. 11**. Общ вид на йерархичен проект във VHDL*

След завършване на VHDL описанието на общата структура студентите могат да тестват проекта, като синтезират и имплементират кода, чрез системата за автоматизирано проектиране Xilinx ISE Design Suite 14.7, която е представена на фиг. 12 и учебните развойни платки с програмируема Digilent Basys2 (фиг. 13).



***Фиг. 12**. САПР Xilinx ISE Design Suite 14.7*



***Фиг. 13**. Развойна платка Digilent Basys2*

ЗАКЛЮЧЕНИЕ

Използването на йерархичния принцип при проектиране с VHDL предлага две съществени предимства: справяне с нарасналата сложност на цифровия хардуер и многократно използване на вече проектирани подсхеми.

В доклада бе представен един учебен проект, следващ принципа на йерархичното проектиране. В проекта се засяга работата на PS/2 интерфейса, изучаван по дисциплината Компютърна периферия. За реализация на представения проект са използвани само стандартни комбинационни и последователностни схеми, които се изучават по дисциплината Анлиз и синтез на логически схеми. В проекта е реализирано динамично управление на 7-сегментни индикатори, което е включено в обучението по дисциплината Микропроцесорна техника. Така с прилагане на йерархичния принцип в обучението по VHDL и автоматизирано проектиране на цифрови схеми се постига: 1)

обединяване и използване по нов начин на знания от вече изучени дисциплини; 2) многократно използване на вече проектирани стандартни модули с еднакви или подобни функции, които са включени в библиотека с VHDL модели или които студентите сами са разработили по дисциплината Автоматизация на инженерния труд; 3) Съвместна работа на двама или трима студента, всеки от които работи по различни модули от проектната йерархия.

REFERENCE

[1] Chu P., RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability. Wiley-IEEE Press, 2006, 694 pages, ISBN 978-0-471-72092-8.

[2] Hwang E., Digital Logic and Microprocessor Design with Interfacing. 2nd edition. Cengang Learning, 2016, ISBN 1305859456.

[3] Kukenska V., P. Minev, I. Varbov. Modeling and Analysis of Interface Module for Keyboard. International Scientific Conference 16 – 17 November 2012, Gabrovo, Vol.1, pp 419–422, ISSN 1313-230X.

[4] Petrov B., P. Minev, I. Varbov. Universal Frequency Divider Based on Programmable Logic Device. 15th Science Conference „Days of Science 2011”, 27 May 2011, V. Tarnovo.