FRI-2G.303-1-CCT1-10

EDUCATIONAL VHDL MODELS OF PROCESSORS WITH VON NEUMANN AND HARVARD ARCHITECTURES[10]

Assoc. Prof. Aneliya Ivanova, PhD
Department of Computing,
"Angel Kanchev" University of Ruse
Phone: 082-888 827
E-mail: aivanova@uni-ruse.bg

Principal Assistant Nikolay Kostadinov, PhD
Department of Computing,
"Angel Kanchev" University of Ruse
E-mail: nkostadinov@uni-ruse.bg

Abstract: *In today's dynamically changing high-tech world, the challenges facing engineering higher education are becoming more and more serious, especially when it comes to training future IT professionals. Under the pressure of rapid changes in the labour market in this field, students come to the university with increasingly high demands and expectations for the quality of the educational process. More urgently than ever, they want to know why they are studying a particular subject from the curriculum and how it relates to their future professional realization. In this sense, the consistency of the curriculum becomes a key factor in the prevention against students' dropout. Building active interdisciplinary connections between the courses should be a mandatory principle when updating the curricula, and the more fundamental a given discipline is, the more important it is to build connections between it and more practically oriented courses. The Design Technology course has always had input connections with its predecessor Computer Organization, but after actualization of the curriculum, these two disciplines are studied in parallel, and this not only poses new challenges, but also uncovers new opportunities for deepening the connections between them. This paper describes an approach to illustrate the principles of operation of two classical computer architectures - von Neumann and Harvard using VHDL tutorial models of processors with such architectures. The aim of the discussed approach consists of the following: after the students have become theoretically familiar with the features of the mentioned above architectures within the Computer Organization course, they will have to develop VDHL models of such processors within the Design Technology course and to experiment with them to consolidate the acquired knowledge. The experiments will be carried out through FPGA implementation of the projects on Digilent's Basys 3 Board.*

Keywords: *Higher education, Interdisciplinary connection, Computer Organization, Design Technology, von Neumann Processor Architecture, Harvard Processor Architecture, VHDL Model, FPGA.*

ВЪВЕДЕНИЕ

В настоящия динамично променящ се високотехнологичен свят предизвикателствата пред инженерното висше образование стават все по-сериозни, особено, когато става дума за подготовка на бъдещи ИТ специалисти. Под натиска на бързите промени в пазара на работна ръка в тази област, студентите идват в университета с все по-високи изисквания и очаквания към качеството на учебния процес. По-настойчиво от всякога те искат да знаят защо изучават дадена дисциплина и каква е връзката й с бъдещата им професионална реализация. В този смисъл консистентността на учебния план става ключов фактор в борбата срещу отпадането на студенти. Изграждането на активни интердисциплинарни връзки между отделните курсове трябва да бъде задължителен критерий при актуализацията на учебните планове и колкото по-фундаментална и основополагаща е дадена дисциплина, толкова по-важно е да са налице активни връзки между нея и по-практически ориентираните курсове.

Дисциплината "Технология на проектирането" винаги е имала входни връзки с предхождащата я "Организация на компютъра", но след промени в учебния план, тези две

[10] Докладът е представен на научната сесия на 27.10.2023 в секция „Комуникационна и компютърна техника" с оригинално заглавие на български език: УЧЕБНИ VHDL МОДЕЛИ НА ПРОЦЕСОРИ С ФОН-НОЙМАНОВА И ХАРВАРДСКА АРХИТЕКТУРА

дисциплини вече се изучават паралелно и това поставя не само нови предизвикателства, но и открива нови възможности за задълбочаване на връзките между тях. Една от тези възможности е да се онагледят принципите на работа на двете класически компютърни архитектури – фон нойманова и харвардска, с помощта на учебни VHDL модели на процесори с такива архитектури. Тези модели могат да бъдат реализирани като проекти, разработени от студентите по време на практическите упражнения по "Технология на проектирането" след като са се запознали в теоретичен план с особеностите на двете архитектури в рамките на курса "Организация на компютъра".

В (Zavala et al., 2015) е описан подход за изучаване на компютърни архитектури чрез разработване на VHDL модел на 8-битов RISC процесор с харвардска архитектура, в (Fouda & Eldeen, 2013) с помощта на VHDL модел се разширява системата инструкции на популярния микроконтролер 8051, в (Nakano & Ito, 2008) и (Larkins et al., 2013) се дискутират подходи за изграждане на връзки между хардуерни и софтуерни дисциплини в учебните планове по компютърни системи чрез разработване на FPGA базирани модели на различни процесори. В последните два примера разработените от студентите проекти служат като база за надграждане в следващи дисциплини, а в другите два посочени примера, реализираните модели имат изолирано, в рамките на курса, приложение.

В настоящия доклад е представен подход за изграждане на паралелни връзки между две едновременно изучавани дисциплини – едната с теоретична, а другата с практическа насоченост, чрез реализиране на VDHL модели на процесори, с които студентите да експерименират и да затвърдят получените теоретични знания относно фон нойманoвата и харвардската архитектури като в процес на проектиране на моделите прилагат и съпътстващи теоретични знания, придобити в дисциплината "Организация на компютъра". Експериментите ще се осъществяват чрез FPGA реализация на проектите с макета Basys 3 Board на Xilinx.

ИЗЛОЖЕНИЕ

Модел на процесор с фон нойманова архитектура

Проектирането на модела на процесора с фон нойманова архитектура преминава през изпълнението на следните стъпки:

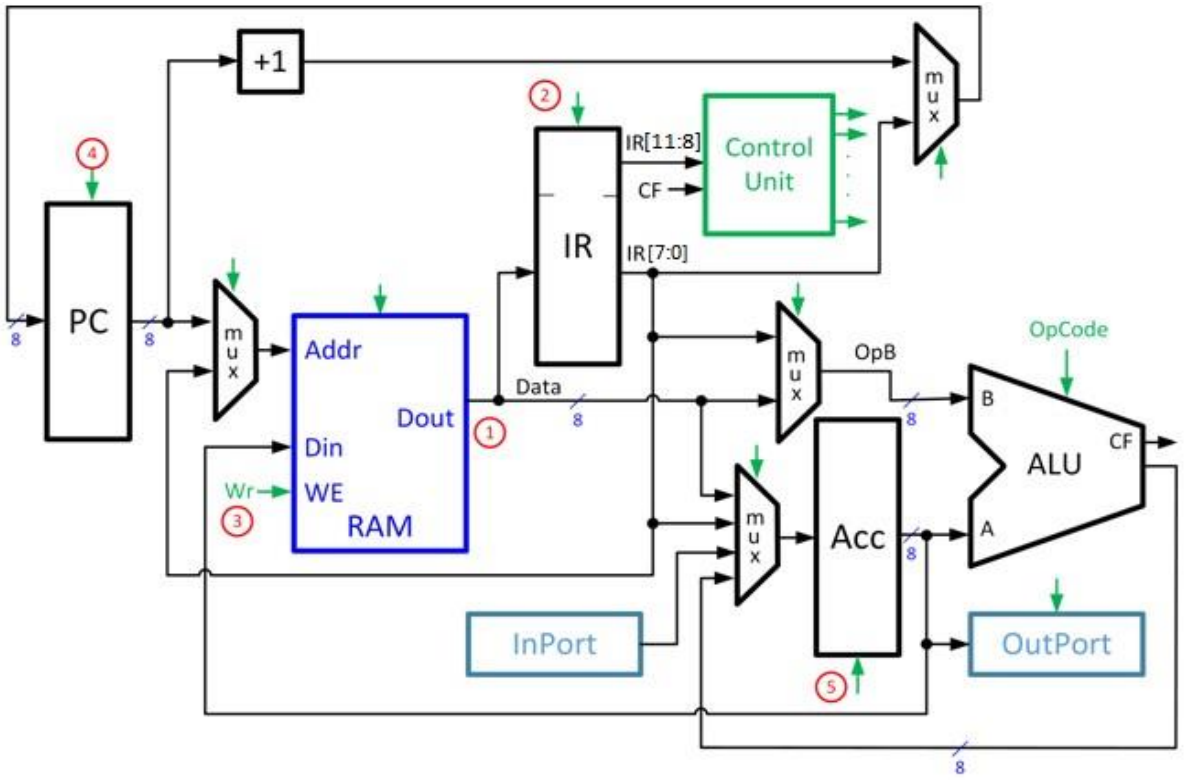
1. Проектиране на системата инструкции (видове инструкции, режими на адресация, формат на инструкциите). Тъй като тези въпроси се разглеждат теоретично по дисциплината "Организация на компютъра", се открива отлична възможнопст да бъдат приложени на практика придобитите знания.
2. Проектиране на операционния блок – аритметико-логическо устройство (АЛУ) с акумулаторна организация, с което ще се затвърдят теоретичните знания относно структурата на АЛУ и организацията му на работа.
3. Проектиране на управляващото устройство, което също е тема, дискутирана в теоретичен план в дисциплината "Организация на компютъра".
4. Спецификация, моделиране и симулация на блоковете на процесора чрез езика за описание на хардуер VHDL.
5. Синтез и реализация на процесора в FPGA схема.

За спецификация, моделиране, симулация и реализация на процесора се използва интегрираната среда за проектиране Xilinx Vivado Design Suite (Vivado, 2023). Реализацията и тестването се извършва в средата на развойния макет Basys 3 (Digilent, 2016), включващ FPGA схема XC7A35T от фамилията Artix-7.

Структурата на проектирания модел на процесор е показана на фиг.1. С оглед предназначението му за учебни цели разрядността, системата инструкции и режимите на адресация са максимално опростени. Процесорът е 8-разряден, с 8-разрядно АЛУ (ALU) като акумулаторът (Acc) е допълнен с девети бит, служещ за флаг за пренос (CF). Регистърът на инструкцията (IR) е 12 разряден, програмният брояч (PC) е 8-разряден, а паметта (RAM) е с обем от 256 клетки и се синтезира на базата на вградения в FPGA компонент от типа

BlockRAM, конфигуриран за конкретния случай като еднопортова памет с организация 256 x 12. За въвеждане и извеждане на данни са предвидени два 8-разрядни порта: входен (InPort) и изходен (OutPort).

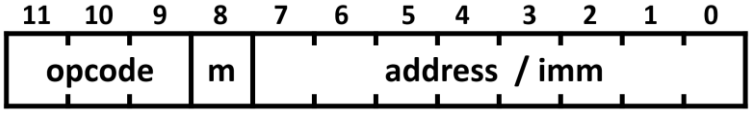
Предвидени са два режима на адресация: абсолютна и непосредствена. Опростената система инструкции (табл.1) включва 7 12-битови инструкции с фиксиран формат (фиг.2). Кодът на операцията (*opcode*) се задава с трите най-старши бита. Режимът на адресация (*m*) се определя от стойността на бит 8 на инструкцията: при 0 с разряди [7:0] се кодира адрес от паметта за данни (*addr*), а при 1 – непосредствен операнд (*imm*). За управление хода на изпълнение на програмата е предвидена инструкция JCC за условен преход при флаг CF=0. Независимо дали преходът е осъществен или не, тази инструкция нулира флага CF, което позволява чрез две последователни инструкции JCC да се извърши безусловен преход.



Фиг. 1. Структурна схема на модела на 8-разряден фон нойманов процесор с акумулаторна организация

Таблица 1. Система инструкции на фон нойматовия процесор

Мнемокод	Синтаксис	Код	Семантика
LOAD	LOAD address	0010[address]	Acc ← mem[address]
	LOAD #imm	0011[imm]	Acc ← imm
STORE	STORE address	0100[address]	mem[address] ← Acc
NOR	NOR address	0110[address]	Acc ← Acc nor mem[address]
	NOR #imm	0111[imm]	Acc ← Acc nor imm
ADD	ADD address	1000[address]	Acc ← Acc + mem[address] (установява CF)
	ADD #imm	1001[imm]	Acc ← Acc + imm (установява CF)
INP	INP	101000000000	Acc ← InPort
OUTP	OUTP	110000000000	OutPort ← Acc
JCC	JCC address	1110[address]	PC ← address при CF=0 (нулира CF)



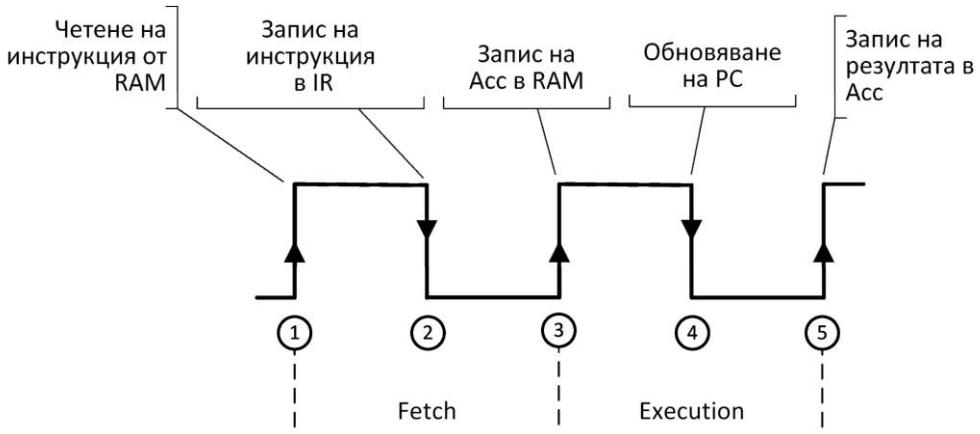
Фиг. 2. Формат на инструкцията

Управляващото устройство се реализира чрез краен автомат с памет като входните сигнали за автомата са най-старшата тетрада на инструкцията (IR[11:8]) и флагът CF. Вътрешното представяне на състоянията е избрано да съответства на полето *opcode* на инструкцията, което опростява реализацията. Така, докато в първата фаза на обработката на инструкцията автоматът е във FETCH_STATE, следващото състояние се определя директно от разряди [11:9] на IR (фиг. 3). Във фазата на изпълнението текущото състояние задава и кода на операцията (OpCode), изпълнявана от ALU.

```
-- Кодиране на състоянията на управляващия автомат
constant FETCH_STATE  : std_logic_vector := "000";
constant LOAD_STATE   : std_logic_vector := "001";
constant STORE_STATE  : std_logic_vector := "010";
constant NOR_STATE    : std_logic_vector := "011";
constant ADD_STATE    : std_logic_vector := "100";
constant INP_STATE    : std_logic_vector := "101";
constant OUTP_STATE   : std_logic_vector := "110";
constant JCC STATE    : std_logic_vector := "111";
-- Машина на състоянията
State <= FETCH_STATE when (Reset = '1' or Clk = '1') else IR(11 downto 9);
```

Фиг. 3. Деклариране и превключване на състоянията на автомата

От времедиаграмата на фиг. 4 е видно, че за изпълнение на една инструкция в този хипотетичен процесор са необходими *два машинни цикъла* като се преминава през следните стъпки: **1)** Извличане на инструкцията от RAM паметта по адреса, съдържащ се в PC. **2)** Запис на прочетената инструкция в IR. Старшите 4 бита на инструкцията се подават към управляващото устройство за декодиране, а младшите 8 в зависимост от адресацията служат за адресиране на паметта или представляват непосредствен операнд. **3)** Тази стъпка се изпълнява само при инструкция STORE, като по преден фронт на тактовия сигнал съдържанието на акумулатора се записва в RAM паметта. **4)** Запис в PC на адреса на следващата инструкция. При инструкция за условен преход JCC и флаг CF=0 това е зададеният в инструкцията адрес, а в останалите случаи – съдържанието на PC, увеличено с 1 (фиг. 5). **5)** Запис на резултата от операцията (инструкции ADD и NOR) в акумулатора. При входно-изходни операции (инструкции INP и OUTP) в тази стъпка се извършва обмен на данни между акумулатора и портовете (фиг.5).



Фиг. 4. Времедиаграма на изпълнение на инструкция във фон нойманова архитектура

```
-- Програмен брояч
PROGRAM_COUNTER:
  process(Clk, Reset)
  begin
    if (Reset = '1') then
      PC <= (others => '0');     -- Стартов адрес: 0
    elsif (Clk'event and Clk = '0') then
      if ((IR(11 downto 9) = JCC_STATE) and (Acc(8) = '0')) then
        PC <= IR(7 downto 0); -- Извършен преход при JCC
      else
        PC <= PC + 1;          -- Следваща инструкция
      end if;
    end if;
  end process PROGRAM_COUNTER;

-- Аритметико-логическо устройство
ALU:
  process(Clk, Reset, OpA, OpB)
  begin
    if (Reset = '1') then
        Acc <= (others => '0');
    elsif (Clk'event and Clk = '1') then
      -- Изпълнение на инструкцията
      case State is
          when LOAD_STATE => Acc(7 downto 0) <= OpB;
          when NOR_STATE => Acc(7 downto 0) <= OpA nor OpB;
          when ADD_STATE => Acc <= ('0' & OpA) + ('0' & OpB);
          when INP_STATE => Acc(7 downto 0) <= InPort;
          when OUTP_STATE => OutPort <= Acc(7 downto 0);
          when JCC_STATE => Acc(8) <= '0';
          when others => null; -- STORE, извличане на инструкция
      end case;
    end if;
  end process ALU;
```

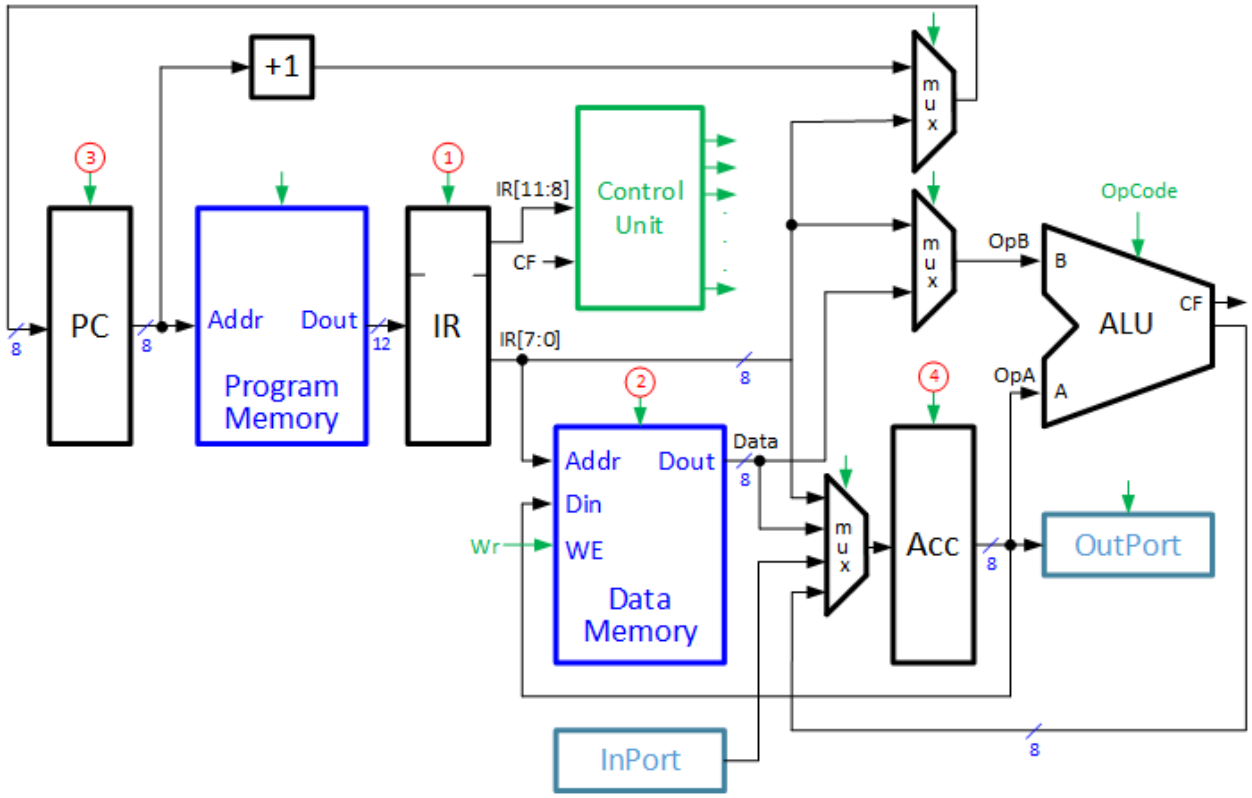
Фиг. 5. Реализация на АЛУ и програмния брояч.

Модел на процесор с харвардска архитектура

Проектирането на този модел на процесор преминава през същите стъпки като разгледаните по-горе, а структурата му (фиг. 6) се отличава от тази на предходния по двете отделни адресни пространства, съответно за команди (Program Memory) и данни (Data Memory). За да могат студентите да направят пълноценно сравнение на двата модела на процесори е целесъобразно последните да имат идентични системи инструкции, видове адресации и формат на инструкцията. Управляващото устройство също се реализира по идентичен начин.

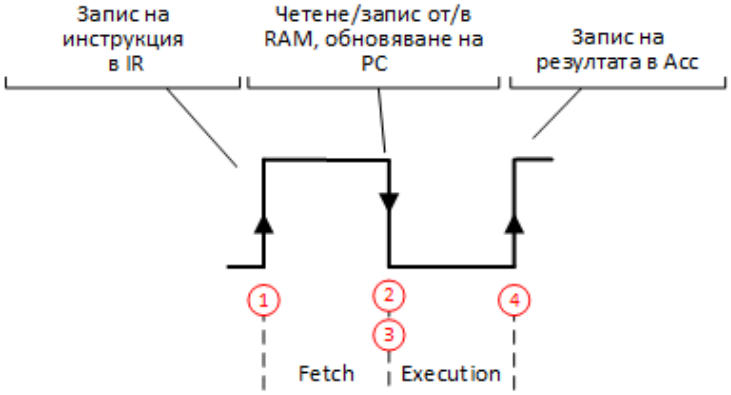
Цикълът на изпълнение на една инструкция (фиг. 7) отразява спецификата на тази архитектура и включва следната последователност от стъпки: **1)** Извличане на инструкцията от програмната памет по адреса, съдържащ се в PC и запис на прочетения код в IR. Старшите 4 бита на инструкцията се подават към управляващото устройство за декодиране, а младшите 8 в зависимост от вида на адресацията служат за адресиране на паметта за данни или играят ролята на непосредствен операнд. **2)** Четене/запис от/в паметта за данни. При абсолютна адресация прочетените данни (Data) представляват втория операнд (OpB) на ALU (инструкции ADD и NOR) или новото съдържание на Acc (инструкция LOAD). В случай на непосредствена адресация към OpB или Acc се мултиплексира младшият байт на инструкцията. След изпълнението на тази стъпка на изхода на ALU се формира резултатът от операцията, като при инструкция ADD се установява и флаг CF. При инструкция STORE в тази стъпка съдържанието на акумулатора се записва в паметта за данни. **3)** Запис в програмния брояч на адреса на следващата инструкция. При инструкция за условен преход JCC и флаг CF=0 това е зададеният в инструкцията адрес, а в останалите случаи – съдържанието на PC, увеличено с 1. **4)** Запис на резултата от операцията (инструкции ADD и

NOR) в акумулатора. При входно-изходни операции (инструкции INP и OUTP) в тази стъпка се извършва обмен на данни между акумулатора и портовете.



Фиг. 6. Структурна схема на модела на 8-разряден процесор с харвардска архитектура и акумулаторна организация

Вградените във FPGA схемата XC7A35T запомнящи блокове, с които могат да се реализират паметите за програми и данни са от синхронен тип, като четенето и записът се извършват по преден фронт на тактовия сигнал. Ако за IR се използва изходният регистър на програмната памет, а тактовият сигнал на паметта за данни е инвертиран, посочените стъпки могат да се изпълнят в рамките на 3 фронта на сигнала Clk. В този случай всяка инструкция се изпълнява за *един машинен цикъл* (фиг. 5) тъй като е налице съвместяване на изпълнението на стъпки 2 и 3 на текущата инструкция по заден фронт на тактовия сигнал, както и на стъпка 4 на текущата инструкция и стъпка 1 на следващата по преден фронт. В това се заключава и най-важният извод, който студентите се очаква да направят след експериментиране и сравнение на работата на двата модела на процесори.



Фиг. 7. Времедиаграма на изпълнение на инструкция в харвардската архитектура

Паметите за данни и програми се синтезират на базата на вградените в FPGA XC7A35T компоненти от типа BlockRAM, конфигурирани за конкретния случай. Паметта за данни

представлява компонент, настроен като RAM с организация 256 x 8, а паметта за програми е компонент, конфигуриран като ROM с обем 256 x 12. Създаването на екземпляри на тези памети и свързването им към сигналите на процесора е отразено на фиг. 8.

```
-- Свързване на RAM
Data_Memory: RAM
  port map(Addr   => IR(7 downto 0),
           Din    => Acc(7 downto 0),
           Dout   => Data,
           WE     => Wr,
           Clk    => not Clk);
-- Свързване на ROM
Program_Memory: ROM
  port map(Addr   => PC,
           Dout   => IR,
           Clk    => Clk);
```

Фиг. 8. Свързване на паметите за данни и програми

ЗАКЛЮЧЕНИЕ

С помощта на предложените учебни модели на процесори се осъществява активна паралелна връзка между практическите упражнения по дисциплината “Технология на проектирането” и няколко важни теми от едновременно преподаваната дисциплина “Организация на компютъра” като Кодиране на числата в компютрите; Структура на процесора; Структура на аритметико-логическото устройство; Видове компютърни архитектури; Система инструкции, формат на инструкциите и видове адресации; Организация на изчислителния процес в компютъра.

Чрез тази форма на активно учене студентите ще задълбочат познанията си по гореспоменатите теми, ще развият уменията си за прилагане на придобитите теоретични знания в решаване на практически проблеми, ще направят една крачка напред в посока асимилиране на замисъла на учебния план и ще получат по-голяма яснота по въпроса защо е необходимо да изучават тези две дисциплини.

ACKNOWLEDGEMENT

This paper is supported by project 23-FEEA-01 “Development of models and simulations with different application areas”, funded by the Research Fund of the University of Ruse.

REFERENCES

Digilent (2016) https://digilent.com/reference/_media/basys3:%20basys3_rm.pdf

Fouda, A.M., A.B.Eldeen. (2013) Design modified architecture for MCS-51 with innovated instructions based on VHDL. Ain Shams Engineering Journal, Vol. 4, Issue 4, pp. 723-733, ISSN 2090-4479, https://doi.org/10.1016/j.asej.2012.12.001.

Larkins, D.B., Jones, W.M., & Rickard, H.E. (2013). *Using FPGAs as a reconfigurable teaching tool throughout CS systems curriculum*. Technical Symposium on Computer Science Education.

Nakano, K. & Y. Ito. (2008). *Processor, Assembler, and Compiler Design Education Using an FPGA*. In Proceedings of the 2008 14th IEEE International Conference on Parallel and Distributed Systems (ICPADS ‘08). IEEE Computer Society, USA, 723–728.

Vivado (2023) https://www.xilinx.com/products/design-tools/vivado.html

Zavala, A.H., O.C. Nieto, J.A. Huerta Ruelas, A.R. Carvallo Domínguez. (2015). *Design of a General Purpose 8-bit RISC Processor for Computer Architecture Learning*. Computación y Sistemas, Vol. 19, No. 2, 2015, pp. 371–385, ISSN 1405-5546, doi: 10.13053/CyS-19-2-1941.