ჩაშენებული სისტემების საიმედოობის გაზრდის ახალი მეთოდების კვლევის შედეგები

გიორგი ჩაჩუა, ია მოსაშვილი
საქართველოს ტექნიკური უნივერსიტეტი

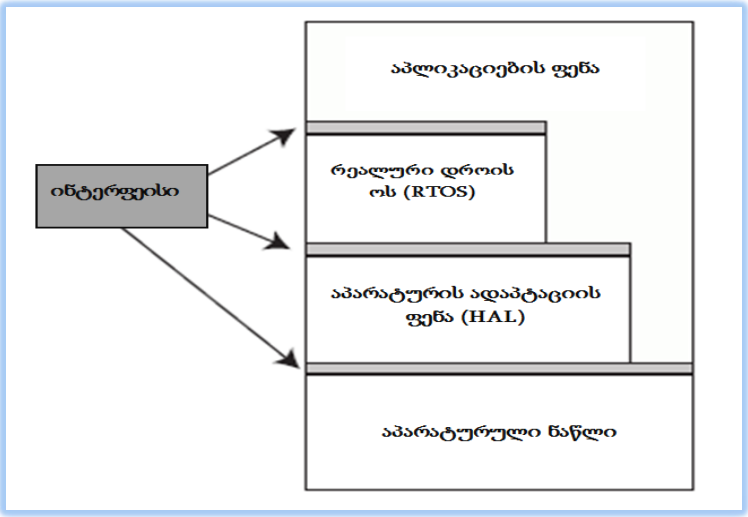
რეზიუმე

განხილულია ჩაშენებული სისტემების არქიტექტურა და მისი შემადგენელი კომპონენტები. დახასიათებულია ვერიფიკაციისა და ტესტირების ის მეთოდები, რომლებიც ყველაზე მეტად მიესადაგება მათი მართვის საიმედოობის გაზრდას და ახალი ტექნოლოგიების შემუშავებას. გამოყენებულია ახალი მეთოდი - „დამუშავება ტესტირების შემდეგ" (Test Driven Development). მოცემულია მისი განხორციელების თითოეული ეტაპი. წარმოდგენილია Spartan FPGA პროგრამირებად დაფაზე შესრულებული კვლევის შედეგები. დაბალი დონის პროგრამირების VHDL-ენაზე შედგენილია ჩაშენებული სისტემების ტესტირების ახალი ბიბლიოთეკები.

საკვანძო სიტყვები: ავტომობილი. ჩაშენებული სისტემა. ტესტირება. Test Driven Development. ავტომობილის საიმედოობა.

1. შესავალი - ჩაშენებული სისტემების არქიტექტურა

ჩაშენებული სისტემა (ჩს), როგორც წესი. განსაზღვრულია კონკრეტული ამოცანების შესასრულებლად, კონკრეტულ გამოთვლით გარემოში, რომელიც შედგება პროგრამული და აპარატურული კომპონენტებისაგან. 1-ელ ნახაზზე გამოსახულია ასეთი სისტემის ტიპური შემადგენლობა, მოცემულია 4 ფენის ტერმინოლოგია, რომელთაგან სამი არის პროგრამული უზრუნველყოფა, ხოლო მეოთხე - აპარატურული [1].



ნახ.1. ჩაშენებული სისტემის სტრუქტურა

2. ჩაშენებული სისტემის ვერიფიკაცია

ვერიფიკაცია პროცესია, რომელიც აჩვენებს, რომ პროგრამა შეესაბამება თავის მახასიათებლებს. თუ მახასიათებლები ან პროგნოზირებადი ტესტები საკმარისად ზუსტი და ინფორმაციულია. შემოწმება (ვერიფიკაცია) შეიძლება იყოს უფრო ობიექტური, ვიდრე ვალიდაცია- რომელიც გამოიყენება მხოლოდ მომხმარებლის მოთხოვნების შესამოწმებლად.

ნაშრომში განხილულია ვერიფიკაციისა და ტესტირების ის მეთოდები, რომლებიც ყველაზე მეტად მიესადაგება ჩაშენებული სისტემების საიმედო ტექნოლოგიების შემუშავებას [2].

3. ჩაშენებული სისტემების ტესტირება

ტესტირება პროცესის განხორციელებაა, რომლითაც ვრწმუნდებით, რომ ის აკმაყოფილებს არსებულ მოთხოვნებს, რათა განვსაზღვროთ განსხვავება მოსალოდნელ და ფაქტობრივ შედეგებს შორის. შემავალი მონაცემები უნდა იყოს მოწოდებული ობიექტური პროგრამისათვის და პროგრამის რეაქცია უნდა იყოს დაკვირვებადი. ფორმალური შემოწმებისაგან განსხვავებით, რომელიც მიმართულია იმისკენ, რომ დაამტკიცოს წინასწარ განსაზღვრული მახასიათებლების შესაბამისობა. ტესტირების მიზანია ისეთი შემთხვევის მოძებნა, როდესაც რეაქცია არ შეესაბამება მოსალოდნელ შედეგს და ჩვენ ასეთ მეთოდს ვუწოდებთ პროგნოზირებად ტესტებს.

ჩაშენებული სისტემის კვლევის დიდი ნაწილი მიმართული იყო დროებითი წყვეტებისკენ, რომლებიც წარმოიშობა მაშინ, როცა სისტემებს აქვს ხისტი მოთხოვნები რეალურ დროში. დროითი გაუმართავობა მნიშვნელოვანია, მაგრამ პრაქტიკაში მისი დიდი პროცენტი ჩაშენებულ სისტემებში მოდის ფუნქციონალურ ქცევასთან, მათ შორის ალგორითმულ ლოგიკურ და კონფიგურაციულ შეცდომებზე და კრიტიკულ სექციებში რესურსების არამიზნობრივ გამოყენებაზე.

ისეთი მიდგომების პოვნა, რომლის მეშვეობითაც უკეთესად შეიძლება მომზადდეს სისტემის კომპონენტებს შორის მიმდინარე ურთიერთზეგავლენის გამოცდა და თვით ამ პროცესის ავტომატიზაცია - პრიორიტეტულია.

ამრიგად, ჩაშენებული სისტემების ტესტირებისა და მართვის საიმედოობის გაზრდისათვის ჩვენს კვლევაში შერჩეული მეთოდია „დამუშავება ტესტირების შემდეგ (Test Driven Development – TDD) [3].

4. დამუშავება ტესტირების შემდეგ

დღესათვის ჩაშენებული სისტემების ვერიფიკაციის ყველაზე ეფექტური მეთოდი არის „დამუშავება ტესტირების შემდეგ" (TDD), რომელიც პროგრამული უზრუნველყოფის დამუშავების ტექნოლოგიაა. იგი ეფუძნება დამუშავების ხანმოკლე ციკლების გამეორებას. თავიდან იწერება ტესტი, რომელიც მოიცავს სასურველ ცვლილებებს, ხოლო შემდეგ იწერება კოდი, რომელიც საშუალებას იძლევა მოხდეს ამ ტესტის გავლა.

ბოლოს ხდება ახალი კოდის შესაბამისი სტანდარტების მიხედვით რეფაქტორინგი. კენტ ბერკი, რომელიც ითვლება ამ ტექნიკის გამომგონებლად, 2003 წელს ამტკიცებდა, რომ „დამუშავება ტესტირების შემდეგ" ამარტივებს ჩაშენებული სისტემების დიზაინს და მატებს მას საიმედოობას [4].

1999 წელს TDD-ს გამოჩენა მჭიდროდ იყო დაკავშირებული კონცეფციასთან – პირველად ტესტი (First Test), რომელიც გამოიყენებოდა ექსტრემალურ პროგრამირებაში, თუმცა შემდგომში გამოიყო როგორც ცალკე მეთოდოლოგია.

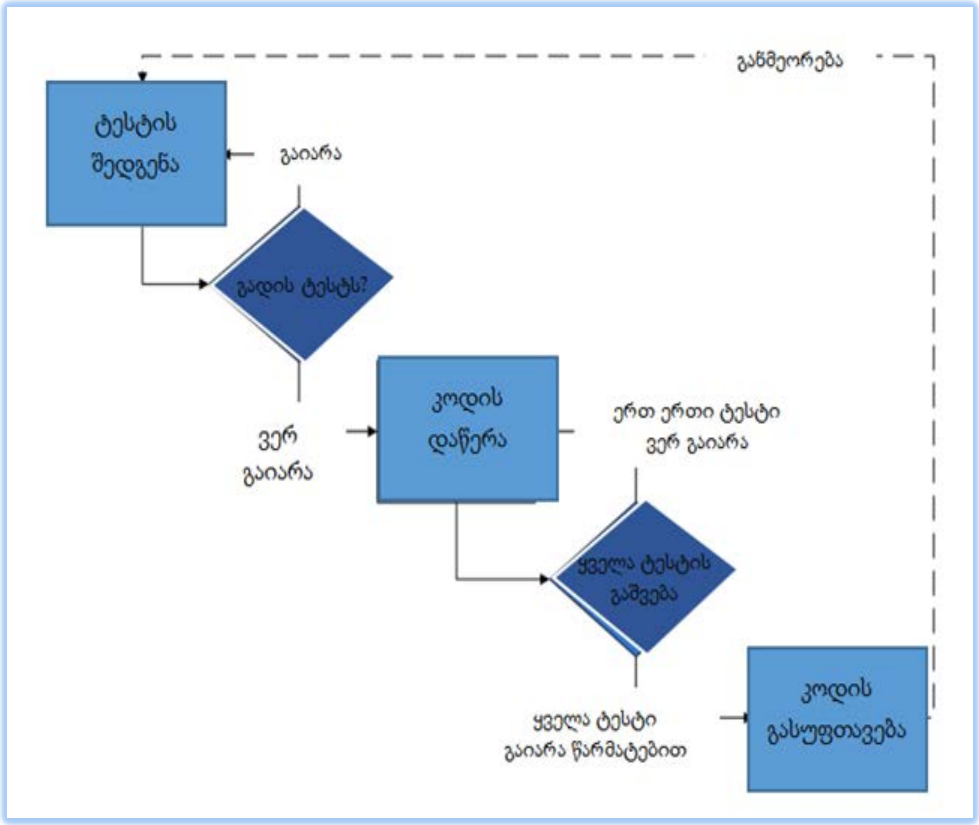
ტესტი არის პროცედურა, რომელიც ადასტურებს ან უარყოფს კოდის ვარგისიანობას, როცა პროგრამისტი ამოწმებს მის მიერ შემოშავებული კოდის ვარგისიანობას, ის ტესტირებას აკეთებს ხელით [5].

TDD მოითხოვს დეველოპერისაგან ავტომატურ მოდულურ ტესტირების შექმნას, რომლებიც განსაზღვრავს მოთხოვნებს კოდის მიმართ, საკუთრივ ამ კოდის დაწერამდე. ტესტი შეიცავს იმ პირობების შემოწმებას, რომელიც შეიძლება შესრულდეს ან არა. როცა ისინი სრულდება – ამბობენ, რომ ტესტი გავლილია. ტესტის გავლა ადასტურებს ქმედებას, რომელიც პროგრამისტის მიერ იყო განსაზღვრული. დეველოპერები ხშირად იყენებენ ტესტირებების ბიბლიოთეკებს (Testing Frameworks), ტესტების შექმნისა და გაშვების ავტომატიზაციისათვის. პრაქტიკაში მოდულური ტესტები ფარავს კოდის კრიტიკულ და არატრივიალურ უბნებს, ეს შეიძლება იყოს კოდი,

რომელიც განიცდის ხშირ ცვლილებებს, კოდი რომლის მუშაობაზეც დამოკიდებულია დიდი რაოდენობით კოდების გამართული მუშაობა და სხვა.

დამუშავების გარემო სწრაფად უნდა რეაგირებდეს კოდის მცირე მოდიფიკაციაზე. პროგრამის არქიტექტურა უნდა ეფუძნებოდეს ერთმანეთის მიმართ ჯაჭვურად სუსტად დაკავშირებულ კომპონენტებს, რის გამოც კოდის ტესტირება მნიშვნელოვნად მარტივდება.

TDD არა მარტო კოდის კორექტულობას ამოწმებს, არამედ ზეგავლენას ახდენს პროგრამის დიზაინზე. ტესტებზე დაყრდნობით დეველოპერები სწრაფად წარმოიდგენენ, თუ რა ფუნქციონალი არის აუცილებელი მომხმარებლისათვის. ამრიგად, ინტერფეისის დეტალები წარმოიქმნება დიზაინის საბოლოო ფაზამდე გაცილებით ადრე.



ნახ.2. TDD-ს ციკლის გრაფიკული წარმოდგენა ბლოკ-სქემების სახით

5. ტესტის დამატება

TDD პროცესში ყველა ახალი ფუნქციონალის დამატება პროგრამაში იწყება ტესტის დაწერით. თავისთავად პროგრამა ამ ტესტს ვერ გაივლის პირველ ეტაპზე, რადგანაც შესაბამისი კოდი ჯერ არ არის დაწერილი (თუ გაივლის ტესტს, ეს იმას ნიშნავს, რომ ეს ფუნქციონალი უკვე გააჩნია ამ პროგრამას ან ტესტს აქვს ნაკლოვანებები). იმისათვის, რომ დაიწეროს ტესტი, დეველოპერს მკაფიოდ უნდა ესმოდეს ახალი ფუნქციონალის მიმართ წაყენებული პირობები. ამისთვის განიხილება გამოყენების განსხვავებული სცენარები და სამომხმარებლო ისტორიები. ახალი მოთხოვნები შეიძლება აგრეთვე გულისხმობდეს არსებული ტესტების მოდიფიკაციას. ეს

განასხვავებს TDD-ს ისეთი ტექნიკისაგან, როცა ტესტები იწერება მას შემდეგ, რაც კოდი უკვე დაწერილია, ის აიძულებს დეველოპერს ფოკუსირება მოახდინოს მოთხოვნებზე კოდის დაწერამდე.

კოდის დაწერა – ამ ეტაპზე იწერება ახალი კოდი ისე, რომ მან გაიაროს ტესტი. ეს კოდი არ არის აუცილებელი იყოს იდეალური. დასაშვებია, რომ მან გაიაროს ტესტი არარელევანტური მეთოდით. ეს მისაღებია, რადგანაც დამუშავების შემდეგ ეტაპებზე ის უფრო გაუმჯობესდება და დაიხვეწება. მნიშვნელოვანია დაიწეროს კოდი, რომელიც განსაზღვრულია მხოლოდ ტესტის გასავლელად, არ არის საჭირო ზედმეტი და შესაბამისად არატესტირებადი ფუნქციონალები.

ტესტების გაშვება – დარწმუნება, რომ პროგრამა ტესტებს გადის. თუ პროგრამა ყველა ტესტს გაივლის, პროგრამისტი დარწმუნებული უნდა იყოს, რომ კოდი აკმაყოფილებს ტესტირების ყველა მოთხოვნას, ამის შემდეგ შეიძლება ციკლის დასკვნით ეტაპზე გადასვლა.

რეფაქტორინგი – როცა მიღწეულია მოთხოვნილი ფუნქციონალი. ამ ეტაპზე კოდი შეიძლება იყოს გაწმენდილი. რეფაქტორინგი არის პროგრამის შიგა სტრუქტურის შეცვლის პროცესი, რომელიც არ ეხება მის გარე ქცევას და მისი მიზანია კოდის მუშაობის შემსუბუქება, კოდის დუბლირების აღმოფხვრა, სამომავლოდ ცვლილების შეტანის გაადვილება და ა.შ. [6].

ციკლის გამეორება – აღწერილი ციკლი მეორდება და ახდენს ახალ-ახალი ფუნქციონალის რეალიზებას, ნაბიჯები უნდა იყოს მცირე შუალედების, თუ ახალი კოდი ვერ აკმაყოფილებს ახალ ტესტებს ან ძველი ტესტები აღარ ფუნქციონრიებს, პროგრამისტმა უნდა მოახდინოს მისი აღმოფხვრა და ცვლილებები შეიტანოს ტესტის ბიბლიოთეკებში, რომელსაც შემდეგ გამოიყენებს ახალი ფუნქციონალის დამატებისას.

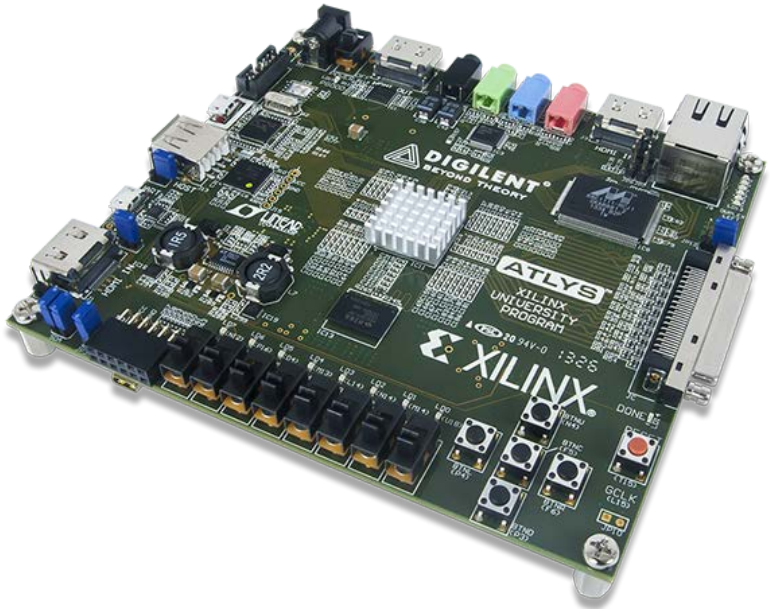
6. დამუშავების სტილი

TDD ხშირად დაკავშირებულია დიზაინის ეტაპის გამარტივების პროცესთან. ითვლება, რომ ამას აქვს ორი უპირატესობა: ის ეხმარება დეველოპერს დარწმუნდეს იმაში, რომ ფუნქციონალი გამოსადეგია ტესტირებისათვის, რადგანაც დეველოპერმა თავიდან უნდა განსაზღვროს, თუ როგორ უნდა მოხდეს მისი ტესტირება. ეს აგრეთვე უზრუნველყოფს იმას, რომ ტესტებით დაფარული იქნება კოდის ყველა ფუნქციონალი.

TDD-ს პირობებში პრაქტიკულმა დამუშავებამ დეველოპერები მიიყვანა ახალი ტექნიკის გამოყენებასთან, რომელსაც ეწოდება დამუშავება მისაღები ტესტირების შემდეგ (Acceptance TDD-ATDD), რომელშიც შემკვეთის კრიტერიუმის აღწერა ავტომატიზირდება ტესტებში და მათ იყენებენ შემდეგ დამუშავების ჩვეულებრივ პროცესში მოდულური ტესტირებისას. ეს პროცესი იძლევა იმის გარანტიას, რომ აპლიკაცია აკმაყოფილებდეს ფორმირებულ მოთხოვნებს. ATDD გამოყენებისას დეველოპერების ჯგუფი ფოკუსირებულია კონკრეტულ ამოცანაზე – დააკმაყოფილოს მისაღები ტესტი, რომელიც ასახავს მომხმარებლის შესაბამის მოთხოვნებს [7].

7. კოდების ნიმუშები VHDL ენაზე

ნაშრომის კვლევების ფარგლებში მიმდინარეობდა ტესტირების ახალი მეთოდის გამოცდა სხვადასხვა მოდიფიკაციის პროგრამირებად FPGA-დაფებზე - DigilenT Spartan 5 FPGA board, Zed board Zynq 7000 FPGA board, N mio rio board (ნახ.3).



ნახ.3. FPGA დაფა.

შეიქმნა ახალი ბიბლიოთეკები, ჩაშენებული სისტემების დაბალი დონის პროგრამირებადი ენის VHDL-ის გამოყენებით. მოხდა მათი ტესტირების შესაბამისი ფაილის შექმნა, შემდგომში მისი სიმულაციურ პროგრამაზე ტესტირება და საბოლოოდ პროგრამირებად დაფაზე იმპლემენტაცია.

8. დასკვნა

TDD გამოყენება ჩაშენებული პროგრამული უზრუნველყოფის სფეროში. იგი დამმუშავებლებს საშუალებას აძლევს შექმნან კარგად ტესტირებული სისტემები. ნაშრომში წარმოდგენილი მეთოდოლოგია არ არის დამოკიდებული ინსტრუმენებსა და პლატფორმაზე, რომლითაც TDD - მეთოდს საშუალებას აძლევს მოახდინოს დამუშავების და ტესტირების რეალიზაცია, რომლითაც იხვეწება პროგრამული უზრუნველყოფის დამუშავება, იზრდება ხარისხი, რასაც საბოლოო ჯამში მივყავართ, საერთო დანახარჯების შემცირებასთან, მოწყობლიობების ექსპლუატაციის პერიოდში მათი დეფექტების შემცირებისა და ფუნქციონალობის მუდმივი დახვეწის შესალებლობების გამო.

დასაწყისში TDD-ს გამოყენება, დაყენება და ინსტრუმენტული საშუალებების შესაბამისობაში მოყვანა მოითხოვს გარკვეულ დროით დანახარჯებს. თუმცა მისი გამოყენებით მიღებული სარგებელი, რამდენჯერმე აღემატება ამ დანაკარგებს. მოსალოდნელია, რომ სრული ავტომატიზაცია ტესტებისა, არასოდეს არ იქნება მიღწეული, თუმცა წარმოდგენილი მეთოდების გამოყენება იძლევა „მოქნილ“ მიდგომას, რომელიც უზრუნველყოფს ტესტირების პროცესის მისაღებ შედეგს.

კვლევის ფარგლებში შექმნილი ბიბლიოთეკები შემდგომში წარმატებით შეიძლება იქნას გამოყენებული, როგორც კვლევების გაგრძელების ახალ მიმართულებებში, ისე პრაქტიკული დანიშნულებითაც, კერძოდ მისი მეშვეობით შესაძლებელია შეიქმნეს ისეთი აპარატურული უზრუნველყოფა, რომელიც პირველ ეტაპზე მოახდენს სამომხმარებლო ბაზარზე არსებული წვრილი და მსხვილი საყოფაცხოვრებო ტექნიკაში ფუნქციონირებადი მართვის ჩაშენებული სისტემის სწრაფ და ეფექტურ დიაგნოსტირებას და პრობლემის აღმოფხვრას.

ლიტერატურა:

1. Schmitt W. (2004). Automated Unit Testing of Embedded ARM Applications. Information Quarterly. Vol.3, N 4, pp.29

2. Clarke E., Garlan D., Krogh B., Simmons R., Wing J. (2000). Verification Tools for Embedded Systems, Carnegie Mellon University, Pittsburgh, PA 15213-3890. Nov.21.

3. Astels D. (2003). Test Driven Development: A Practical Guide, Upper Saddle River, NJ: Prentice Hall PTR

4. https://en.wikipedia.org/wiki/Test-driven_development

5. Unit Test. http://en.wikipedia.org/wiki/Unit_test.

6. Gamma E., Helm R., Johnson R., Vlissides J. (1995). Design Patterns: Elements of Reusable Object-Oriented Software, Reading, MA: Addison-Wesley Professional Computing Series.

7. სურგულაძე გ., გულიტაშვილი მ., კივილაძე ნ. (2015). Web-აპლიკაციების ტესტირება, ვალიდაცია და ვერიფიკაცია. სტუ, „IT-კონსალტინგის ცენტრი“. თბ., -205გვ. http://gtu.ge/book-/gia_sueguladze/4%20GiaSurg%20WebSoftwareTesting.pdf

RESEARCH RESULTS OF NEW METHODS TO IMPROVE THE RELIABILITY OF EMBEDDED SYSTEMS

Chachua Giorgi, Mosashvili Ia
Georgian Technical University.

Summary

This article discusses the architecture and components of embedded systems (ES). Characterized by the methods of inspection and testing of the Armed Forces, which are the most suitable for their management in order to improve the reliability and the development of new technologies. For that in our study we used a new method - Test Driven Development - TDD. The paper presents each stage of the implementation of TDD. Given the results of studies on programmable FPGA Spartan board and on VHDL programming language was created new libraries, which can then be successfully used as a practical purposes, as well as in research in new directions.

НАУЧНЫЕ РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЯ НОВЫХ МЕТОДОВ ДЛЯ ПОВЫШЕНИЯ НАДЕЖНОСТИ ВСТРОЕННЫХ СИСТЕМ

Чачуа Г., Мосашвили И.
Грузинский Технический Университет

Резюме

Рассматриваются архитектура и компоненты встроенных систем (ВС). характеризуется методы проверки и тестирования ВС, которые являются наиболее подходящими для их управления с целью повышения надежности и развития новых технологий. для этого в нашем исследовании использовалась новая методика - Разработка через тестирование - TDD. В Работе приведено каждый этап реализации TDD. Даны итоги исследований на программируемом плате FPGA Spartan и на языке программирования VHDL создано новые библиотеки, которые впоследствии могут быть успешно использованы в качестве практических целей, а также в исследовании в новых направлениях.