Інтэрфейсы: модуль SIM5360E кіруецца сістэмай АТ-каманд праз фізічныя інтэрфейсы UART (да 4 Мбіт/с) або USB (high speed 480 Мбіт/с). UART-інтэрфейс модуля SIM5360E падтрымлівае поўны набор стандартных вывадаў: RXD, TXD, RTS, CTS, DTR, DCD і RI: RXD і TXD – для двухбаковага абмену паміж модулем і вонкавым кіравальным хостам; RTS і CTS – для апаратнага кантролю патокам; DTR – для кіравання бягучым злучэннем; DCD – як індыкатар стану злучэнняў; RI – індыкатар уваходных галасавых выклікаў і SMS-паведамленняў. Для стварэння базы даных выкарыстоўвалася MySQL – рэляцыйная сістэма кіравання базамі даных, якая адносіцца да свабоднага ПЗ.

Прылада звяртаецца да сервера, пасылаючы GET-запыт, які змяшчае вымераныя даныя.

ПЗ сервера складаецца з трох асноўных скрыптоў:

1) set.php – скрыпт для прыёму і апрацоўкі запыту;

2) result.php – скрыпт для адлюстравання вынікаў у выглядзе графікаў;

3) admin.php – скрыпт для кіравання прыладамі.

Скрыпт апрацоўкі запыту. Сервер, атрымаўшы дадзены запыт, апрацоўвае яго, вылучаючы значэнні пераменных, бярэ бягучы час сервера, і ўсе гэтыя значэнні змяшчае ў табліцу «Даныя».

Скрыпт прагляду даных. Пасля аўтарызацыі на дадзенай старонцы будзе пабудаваны графік з вынікамі вымярэнняў, з магчымасцю выбару пачатку дыяпазону, канца дыяпазону, выбару прылады (толькі тых, якія лічацца за гэтым карыстальнікам) і параметраў, якія маюць быць адлюстраваны.

У заключэнне варта адзначыць, што рэалізацыя HTTP кліента рэкамендуецца для прылад, асноўнае прызначэнне якіх – збор даных і перадача іх на сервер. Да недахопаў рэалізацыі HTTP кліента можна аднесці немагчымасць аператыўнага кіравання прыладай. Кіравальныя каманды перадаюцца серверам на прыладу ў адказе на запыт кліента.

Калі ж сістэма прызначана для кіравання рознымі выканаўчымі прыладамі, то неабходна рэалізаваць HTTP сервер. У гэтым выпадку дыспетчар можа аператыўна кіраваць прыладамі.

Літаратура

1. Защита трубопроводов от коррозии : в 2 т. / Ф. М. Мустафин [и др.]. – СПб. : Недра, 2007. – Т. 2. – 708 с.

ПРАГРАМАВАННЕ ПЛІС З ДАПАМОГАЙ ВЫСОКАЎЗРОЎНЕВАЙ МОВЫ ПРАГРАМАВАННЯ

І. Р. Кажамякін

Установа адукацыі «Гомельскі дзяржаўны тэхнічны ўніверсітэт імя П. В. Сухога», Рэспубліка Беларусь

Навуковы кіраўнік В. А. Хананаў

У дадзенай рабоце разглядаецца рэалізацыя прыкладных алгарытмаў на FPGA.

Праграмаваная лагічная інтэгральная схема (ПЛІС) – электронны кампанент (інтэгральная мікрасхема), які выкарыстоўваецца для стварэння канфігуруемых лічбавых электронных схем. У адрозненне ад звычайных лічбавых мікрасхем логіка работы ПЛІС не вызначаецца пры вырабе, а задаецца з дапамогай праграмавання (праектавання). Для праграмавання выкарыстоўваюцца праграматар і IDE (адладка-

вае асяроддзе), якія дазваляюць задаць жаданую структуру лічбавай прылады ў выглядзе прынцыповай электрычнай схемы або праграмы на спецыяльных мовах апісання апаратуры: Verilog, VHDL, AHDL і інш.

ПЛІС шырока выкарыстоўваецца для пабудовы розных па складанасці і па магчымасцях лічбавых прылад:

• з вялікай колькасцю партоў ўводу-вываду (бываюць ПЛІС з больш чым 1000 вывадамі («пінамі»));

• выконваючых лічбавую апрацоўку сігналу (ЛАС);

• лічбавай відэа-, аўдыёапаратуры;

• прылад, якія выконваюць перадачу даных на высокай хуткасці;

• выконваючых крыптаграфічныя аперацыі, сістэм абароны інфармацыі;

• прылад, прызначаных для праектавання і прататыпавання інтэгральных схем спецыяльнага прызначэння (ASIC);

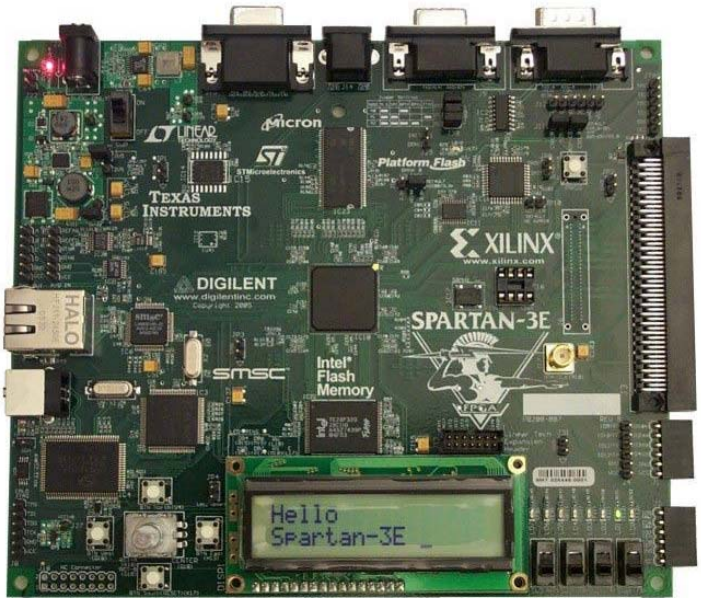
• выконваючых ролю мастоў (камутатараў) паміж сістэмамі з рознай логікай і напружаннем сілкававання;

• рэалізацыі нейрачыпаў;

• прылад, якія выконваюць мадэляванне квантавых вылічэнняў;

• выконваючых апрацоўку радыёлакацыйнай інфармацыі.

Знешні выгляд платы Spartan-3E Starter Kit дадзены на мал. 1.



Мал. 1. Знешні выгляд платы Spartan-3E Starter Kit

Плата змяшчае ў сваім складзе наступныя асноўныя элементы:

• FPGAXC3S500E сямейства Spartan-3E у корпусе FG320;

• ППЗП серыі Platform Flash XCF04S, прызначаная для захоўвання канфігурацыйных даных FPGAXC3S500E;

• блок загрузкі канфігурацыйных даных;

• схема кіравання канфігураваннем FPGA;

• дапаможная FPGA CPLD XC2C64A сямейства CoolRunner-II;

• паслядоўная ППЗП EEPROM, якая падтрымлівае крыптаграфічны алгарытм SHA-1;

• блок сінхранізацыі, прызначаны для фарміравання знешніх (у дачыненні да FPGA) тактавых сігналаў;

• знешняя высокахуткасная АЗП;

• вузел двухканальнага аналага-лічбавага пераўтваральніка (АЛП);

• вузел чатырохканальнага лічба-аналагавага пераўтваральніка (ЛАП);

• модуль паралельнай NOR Flash-памяці ёмістасцю 16 Мбайт;

• модуль паслядоўнай Flash-памяці аб'ёмам 16 Мбіт з інтэрфейсам SPI;

• двухрадковы вадкакрысталічны дысплей на 16 знакамесцаў;

• схема пераўтварэння ўзроўняў сігналаў інтэрфейсу RS-232;

• стандартныя раздыманні інтэрфейсаў RS-232, PS/2, VGA, Ethernet.

ПЛІС у большасці сваёй праграмуюцца на спецыяльных мовах апісання апаратуры: Verilog, VHDL, ADHL і інш. Але таксама існуе магчымасць напісання коду пры карыстанні высокаўзроўневай мовы праграмавання.

У якасці высокаўзроўневай мовы праграмавання будзе выкарыстоўвацца Python – мова праграмавання агульнага прызначэння з дынамічнай строгай тыпізацыі і аўтаматычным кіраваннем памяццю; арыентаваны на павышэнне прадукцыйнасці распрацоўніка, чытальнасці кода і яго якасці, а таксама на забеспячэнне пераноснасці напісаных на ім праграм.

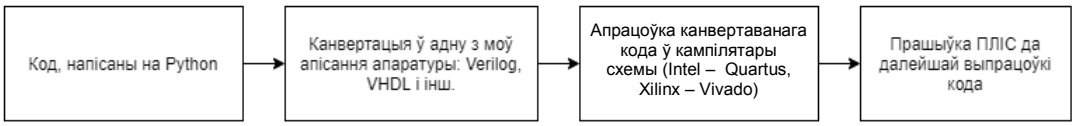
Ёсць некалькі прычын, па якім выкарыстоўваецца гэтая мова:

– з’яўляецца сусветна вядомай;

– нескладаная ў вывучэнні;

– мае неабходны фрэймворк «MyHDL».

Алагрытм стварэння праграмы з дапамогай высокаўзроўневай мовы праграмавання прадстаўлен на мал. 2.



Мал. 2. Алагрытм стварэння праграмы з дапамогай высокаўзроўневай мовы праграмавання

Прыклад праграмы, якая запальвае святлодыёды пры націску на кнопку:

• Python:

```
from myhdl import *
from random import randrange
defled_blinker(input1, led1, led2, led3, led4):
   @always_comb
defon_off_led():
      if input1 == 1:
         led1.next = 1
         led2.next = 1
         led3.next = 0
         led4.next = 0
      else:
         led1.next = 0
         led2.next = 0
         led3.next = 1
```

```
            led4.next = 1
    return on_off_led
```

• Канвертацыя ў Verilog:

```
def convert():
    input1, led1, led2, led3, led4 = [Signal(bool(0)) for i in range(5)]
    toVerilog(led_blinker, input1, led1, led2, led3, led4)
convert()
```

• Вынік канвертацыі – Verilog код:

```
module led_blinker (
    input1,
    led1,
    led2,
    led3,
    led4
);
input input1;
output led1;
reg led1;
output led2;
reg led2;
output led3;
reg led3;
output led4;
reg led4;
always @(input1) begin: LED_BLINKER_ON_OFF_LED
    if ((input1 == 1)) begin
        led1 = 1;
        led2 = 1;
        led3 = 0;
        led4 = 0;
    end
    else begin
        led1 = 0;
        led2 = 0;
        led3 = 1;
        led4 = 1;
    end
end
endmodule
```

Пасля гэтагатрэба передаць Verilog-файл кампілятару і перанесці праграму на інструментальную платформу.

Літаратура

1. ПЛИС. – Режим доступа: https://ru.wikipedia.org/wiki/ПЛИС. – Дата доступа: 16.03.2021.
2. Python. – Режим доступа: https://ru.wikipedia.org/wiki/Python. – Дата доступа: 16.03.2021.
3. FPGA. Разбираемся, как устроены программируемые логические схемы и чем они хороши. – Режим доступа: https://xakep.ru/2018/11/15/fpga/. – Дата доступа: 16.06.2021.
4. Сделай шаг к ПЛИС. – Режим доступа: https://habr.com/ru/post/274829/. – Дата доступа: 16.03.2021.