НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ І ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №6

з дисципліни **«**Технології проектування комп’ютерних систем**»**

Виконав:

студент групи ІО-53, ФІОТ

Гріщенко Костянтин Сергійович

Перевірив:

Сергієнко Анатолій Михайлович

Київ - 2018 р.

**Тема:** Арифметичний пристрій.

**Мета та основні завдання** **роботи**: оволодіти знаннями і практичними навичками з проектування обчислювальних блоків, таких як арифметичний пристрій (AU). Лабораторна робота також служить для оволодіння навичками програмування і налагодження опису AU на мові VHDL.

**Завдання на лабораторну роботу:** розробити арифметичний пристрій за наведеними нижче умовами:

Забезпечити виконання наступних операцій на AU:

|  |
| --- |
| ADD |
| SUB |
| AND |
| NOT |
| ABS |
| SRL |
| MUL |

До складу AU включити наступні основні частини:

|  |  |
| --- | --- |
| Складова частина: | Характеристики: |
| Блок пам’яті | Тип – регістровий, FM.  Розрядність слів – 24 біт.  Кількість слів – 64.  Кількість каналів – 3.  Заборонено запис в перший регістр. |
| LSM | Операції: ADD, SUB, AND.  Розрядність операндів – 24. |
| MPU | Операція: MUL.  Розрядність операндів – 24.  Розрядність результату – 48.  Розрядність вихідної шини – 24. |

Виконати описання поведінкової моделі. Забезпечити контроль бітів переповнення в результаті, знаку результату, ознаки нульового результату. Провести аналіз отриманих графіків роботи схем.

**Хід проектування:**

***Виконаємо опис та кодування операцій:***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Операція** | **Мнемоніка** | **Код операції ACOP** | **Код LSM** | **Біт переповнення** |
| Сума | ADD | 000 | 00 | + |
| Різниця | SUB | 011 | 11 | + |
| Логічне «І» | AND | 001 | 01 | - |
| Логічне «АБО» | OR | 010 | 10 | - |
| Заперечення | NOT | 100 | - | - |
| Модуль | ABS | 101 | - | - |
| Зсув вправо логічний | SRL | 110 | - | - |
| Множення | MUL | 111 | - | - |

*Операція «АБО» була додана для повноти використання всіх розрядів трьохбітового кодування.*

**Виконання:**

***Поведінкова модель:***

***MPU***

entity MPU is

port(C : in BIT;

RST : in BIT;

START:in BIT;

OUTHL:in BIT;

DA : in BIT\_VECTOR(23 downto 0);

DB : in BIT\_VECTOR(23 downto 0);

RDY : out BIT;

Z: out BIT;

N: out BIT;

DP : out BIT\_VECTOR(23 downto 0) );

end MPU;

architecture BEH of MPU is

component OB is port(

C : in BIT; --synchro

RST : in BIT; --reset

LAB : in BIT; -- load A,B, reset P

SHIFT : in BIT; --shift A and B flag

OUTHL : in BIT; --get first(1) or last(0) result word

DA : in BIT\_VECTOR(23 downto 0); --A bus

DB : in BIT\_VECTOR(23 downto 0); --B bus

ADD : in BIT; --adder start flag

REZ : in BIT; --correction flag

B0 : out BIT; --first bit B

STOP : out BIT; --stop flag

Z: out BIT; -- result zero flag

N: out BIT; -- result sign

DP : out BIT\_VECTOR(23 downto 0)); -- result bus

end component;

component FSM is port(

C : in BIT; --synchro

RST : in BIT; --reset FSM

START : in BIT; --start FSM

B0 : in BIT; --first bit B flag

STOP: in BIT; -- check '1' bits of B

LAB : out BIT; -- load operands

SHIFT : out BIT; --shift A and B

ADD: out BIT; -- add

REZ: out BIT; --correction flag

RDY : out BIT); --finish

end component ;

signal lab,shift,add,b0,stop,rez:bit;

begin

--operation block

U\_OP:OB port map(C,RST,

LAB=>lab, SHIFT=>shift,

ADD=>add,B0=>b0,OUTHL=>OUTHL,

DA=>DA, DB=>DB, STOP=>stop, REZ=>rez,

Z=>Z,N=>N,DP=>DP);

--final state machine

U\_FSM:FSM port map(C,RST, -- ??????????? ???????

START=>start, B0=>b0, LAB=>lab, STOP=>stop, REZ=>rez,

ADD=>add, SHIFT=>shift, RDY=>RDY);

end BEH;

**LSM**

library ieee;

use ieee.numeric\_bit.all;

library cnetlist;

use cnetlist.all;

entity LSM is

port(F : in BIT\_VECTOR(1 downto 0);-- function

A : in BIT\_VECTOR(23 downto 0);-- first operand

B : in BIT\_VECTOR(23 downto 0);-- second operand

Y : out BIT\_VECTOR(23 downto 0);-- result

C15: out BIT; -- transfer output

Z: out BIT; -- bit of zero result

N: out BIT);-- sign bit

end LSM;

architecture BEH of LSM is

signal atcc, btcc, ytcc : BIT\_VECTOR(24 downto 0);

signal uatcc,ubtcc,uytcc : BIT\_VECTOR(23 downto 0);

signal ai, bi, yi: integer;

signal sb: BIT;

signal usa,usb : unsigned(23 downto 0);

signal uy: unsigned (24 downto 0);

signal yb: BIT\_VECTOR (24 downto 0);

begin

--for SUB--

SUB:sb<=not B(23) when F="11" else B(23) when F="00";

--two's complement code of operands--

INV\_A: uatcc(22 downto 0)<=not A(22 downto 0) when A(23)='1' else A(22 downto 0) when A(23)='0';

INV\_B: ubtcc(22 downto 0)<=not B(22 downto 0) when sb='1' else B(22 downto 0) when sb='0';

INC\_A: atcc(22 downto 0)<=cnetwork.INT\_TO\_BIT(cnetwork.BIT\_TO\_INT(uatcc(22 downto 0))+1 , 23) when A(23)='1' else uatcc(22 downto 0) when A(23)='0';

INC\_B: btcc(22 downto 0)<=cnetwork.INT\_TO\_BIT(cnetwork.BIT\_TO\_INT(ubtcc(22 downto 0))+1 , 23) when sb='1' else ubtcc(22 downto 0) when sb='0';

SIGN\_A: atcc(23)<=A(23);

SIGN\_B: btcc(23)<=sb;

-- Adder --

ADDER: uy(24 downto 0) <= RESIZE(unsigned(atcc(24 downto 0)),25) + RESIZE(unsigned(btcc(23 downto 0)),17) when (F = "00" or F = "11");

-- Result correction --

INV\_Y:uytcc(22 downto 0) <= not BIT\_VECTOR(uy(22 downto 0)) when uy(23)='1' else BIT\_VECTOR(uy(22 downto 0));

INC\_Y:yb(23 downto 0) <= cnetwork.INT\_TO\_BIT(cnetwork.BIT\_TO\_INT(uytcc(23 downto 0))+1 , 23) when uy(23)='1' else uytcc(23 downto 0);

SIGN\_Y:yb(24)<=BIT(uy(24));

-- Results multiplexer --

MUX: with F select

Y <= yb(23 downto 0) when "00"|"11", -- adder

(A and B) when "01", --AND

(A or B) when "10"; --OR

C15 <= not BIT(uy(24)) when BIT(uy(23))='0' else BIT(uy(24)); -- output transfer

N <= yb(15); --sign bit

ZERO\_DEF: Z <= '1' when yb (23 downto 0) = X"000000" else '0'; -- zero sign

end BEH;

***AU***

entity AU is port(

C : in BIT; --sycnhro

RST : in BIT; --reset

START : in BIT; --start operation in AU

RD: in BIT; -- read from FM to DO bus

WRD : in BIT; -- write from DI bus

RET : in BIT; -- return from subprogram

CALL: in BIT; -- start subprogram

DI : in BIT\_VECTOR(23 downto 0); --in data bus

DO : out BIT\_VECTOR(23 downto 0); --out data bus

AB : in BIT\_VECTOR(5 downto 0); -- register B address

AD : in BIT\_VECTOR(5 downto 0); -- register D address

AQ : in BIT\_VECTOR(5 downto 0); -- register Q address

ARET : in BIT\_VECTOR(20 downto 0); --in return address

ACOP : in BIT\_VECTOR(2 downto 0); -- AU operation code

RDY : out BIT; --result ready

ARETO : out BIT\_VECTOR(20 downto 0);--out return address

BO : out BIT\_VECTOR(23 downto 0); --out data bus D

CNZ: out BIT\_VECTOR(2 downto 0)); --out state reg

end AU;

architecture BEH of AU is

component FM is port(

C: in BIT; -- synchro

WR: in BIT; -- write

INCQ: in BIT;-- inc Q address

CALL: in BIT;-- write return address and CNZ

AB: in BIT\_VECTOR(5 downto 0);-- address B

AD: in BIT\_VECTOR(5 downto 0);-- address D

AQ: in BIT\_VECTOR(5 downto 0);-- address Q

ARETC: in BIT\_VECTOR (23 downto 0);--????? ???????? ? CNZ

Q: in BIT\_VECTOR (23 downto 0);-- ?????? ?????? Q

B: out BIT\_VECTOR(23 downto 0);-- ?????? ?????? ?

D: out BIT\_VECTOR(23 downto 0);-- ?????? ?????? D

ARETCO: out BIT\_VECTOR (23 downto 0));-- ????? ???????? ? CNZ

end component ;

component LSM is port(

F : in BIT\_VECTOR(1 downto 0);-- function

A : in BIT\_VECTOR(23 downto 0);-- first operand

B : in BIT\_VECTOR(23 downto 0);-- second operand

Y : out BIT\_VECTOR(23 downto 0);-- result

C15: out BIT; -- transfer output

Z: out BIT; -- bit of zero result

N: out BIT);-- sign bit

end component ;

component MPU is port(

C : in BIT;

RST : in BIT;

START:in BIT;

OUTHL:in BIT;

DA : in BIT\_VECTOR(23 downto 0);

DB : in BIT\_VECTOR(23 downto 0);

RDY : out BIT;

Z: out BIT;

N: out BIT;

DP : out BIT\_VECTOR(23 downto 0) );

end component ;

type STAT\_AU is (free,mpy,mpyl);-- fsm states

signal st:STAT\_AU;

signal b,q,d,y,dp,aretc,aretco:BIT\_VECTOR(23 downto 0);

signal c0,c15,csh,zlsm,wr,mult,outhl:BIT;

signal rdym,zmpy,nmpy:BIT;

signal cnzr,cnzo,cnzi:BIT\_VECTOR(2 downto 0);

begin

---------FM unit---------

U\_FM: FM port map(C,

WR=>wr, INCQ=>outhl, CALL=>CALL,

AB=>AB, AD=>AD, AQ=>AQ,

ARETC=>aretc,

Q=>q, B=>b, D=>d,

ARETCO=>aretco);

aretc<=cnzr&ARET;

cnzo<=aretco(23 downto 21); --read control bits (C, N, Z) from high bits of last register of FM

ARETO<=aretco(20 downto 0); --read return address from low bits of register of FM

--MUX\_C:c0<='1' when ACOP(1 downto 0)="10" else --????????????? ?0

-- cnzi(2) when ACOP(1 downto 0)="01" else '0';

----------LSM unit----------

U\_LSM:LSM port map(F=>ACOP(1 downto 0),

A=>d,B=>b,Y =>y,

C15=>c15, Z =>zlsm );

----------MPU----------

U\_MPU:MPU port map(C,RST,

START=>mult, OUTHL=>outhl,

DA=>d, DB=>b,

RDY=>rdym,Z=>zmpy,

N=>nmpy, DP=>dp);

---------MUL multiplexor---------

--MUX\_CI:csh<=cnzi(2) when ACOP(1 downto 0)="01" else --??????. ????.

-- cnzi(1) when ACOP(1 downto 0)="11" else '0';

----- result multiplexor ---------

MUX\_Q: q<=dp when st/=free else --multiply result

'0'&d(23 downto 1) when ACOP="110" else --logical right shift

'0'&d(22 downto 0) when ACOP="101" else --abs

not d(23 downto 0) when ACOP="100" else --not

DI when WRD='1' else --result subprogram

d when RD='1' else --data from AD in AD address

y; --LSM result

-------state refister with multiplexor--------------

SR:process(C,RST)

begin

if RST='1' then

cnzi<="000";

elsif C='1' and C'event then

if RET='1' then

cnzi<=cnzo;

elsif st=mpyl then

cnzi<='0'&nmpy&zmpy;

elsif mult='0' then

cnzi<=c15&y(15)&zlsm;

end if;

end if;

end process;

mult<='1' when ACOP="111" else '0';--multiply decoder

---------AU FSM-----------

FSM\_AU:process(C,RST)

begin

if RST='1' then

st<=free; -- fsm state register

elsif C='1' and C'event then

case st is

when free => if START='1'and mult='1'then --not use

st<=mpy;

end if;

when mpy=> if rdym='1' then -- there is a multiplication

st<=mpyl ;

end if;

when mpyl=> st<=free; --multiplication finish

end case;

end if;

end process;

--fsm outs

outhl<='1' when st=mpyl else '0';

wr<='1' when WRD='1' or st=mpyl or (st=mpy and rdym='1') or (START='1' and mult='0') else '0';

RDY<='1'when st=mpyl or (WRD='0' and st/=mpy and mult='0') else'0';

DO<=q; --output data

BO<=B;

CNZ<=cnzi; --state reg out

end BEH;

***FM***

Library IEEE;

use IEEE.NUMERIC\_BIT.all;

entity FM is port(C: in BIT; -- synchro

WR: in BIT; -- write

INCQ: in BIT;-- inc Q address

CALL: in BIT;-- write return address and CNZ

AB: in BIT\_VECTOR(5 downto 0);-- address B

AD: in BIT\_VECTOR(5 downto 0);-- address D

AQ: in BIT\_VECTOR(5 downto 0);-- address Q

ARETC: in BIT\_VECTOR (23 downto 0);--in address CNZ

Q: in BIT\_VECTOR (23 downto 0);-- data Q

B: out BIT\_VECTOR(23 downto 0);-- data B

D: out BIT\_VECTOR(23 downto 0);-- data D

ARETCO: out BIT\_VECTOR (23 downto 0));--out address CNZ

end FM;

architecture BEH of FM is

type MEM64X16 is array(0 to 63) of BIT\_VECTOR(23 downto 0);

constant FM\_init: MEM64X16:= -- initial memory status

(others=> X"000000");

signal RAM: MEM64x16:=FM\_init;

begin

FM16:process(C,AD,AB)

variable addrq,addrd,addrb:natural;

begin

addrq:=TO\_INTEGER(UNSIGNED(AQ));

if INCQ='1' then

addrq:=addrq+1;

end if;

addrd:=TO\_INTEGER(UNSIGNED(AD));

addrb:=TO\_INTEGER(UNSIGNED(AB));

if C='1' and C'event then

if WR = '1' and (addrq /= 0) then

RAM(addrq)<= Q;

end if;

if CALL = '1' then

RAM(63)<= ARETC;

end if;

end if;

B<= RAM(addrb);

D<= RAM(addrd);

ARETCO<= RAM(63);

end process;

end BEH;

**Тестовий стенд**

Library IEEE;

use IEEE.NUMERIC\_BIT.all;

entity au\_tb is

end au\_tb;

architecture TB\_ARCHITECTURE of au\_tb is

component AU is port(

C : in BIT; --sycnhro

RST : in BIT; --reset

START : in BIT; --start operation in AU

RD: in BIT; -- read from FM to DO bus

WRD : in BIT; -- write from DI bus

RET : in BIT; -- return from subprogram

CALL: in BIT; -- start subprogram

DI : in BIT\_VECTOR(23 downto 0); --in data bus

AB : in BIT\_VECTOR(5 downto 0); -- register B address

AD : in BIT\_VECTOR(5 downto 0); -- register D address

AQ : in BIT\_VECTOR(5 downto 0); -- register Q address

ARET : in BIT\_VECTOR(20 downto 0); --in return address

ACOP : in BIT\_VECTOR(2 downto 0); -- AU operation code

RDY : out BIT; --result ready

ARETO : out BIT\_VECTOR(20 downto 0);--out return address

DO : out BIT\_VECTOR(23 downto 0); --out data bus

BO : out BIT\_VECTOR(23 downto 0); --out data bus D

CNZ: out BIT\_VECTOR(2 downto 0)); --out state reg

end component ;

signal c,rst,rdy,start,wrd,rd,ret,call:BIT;

signal acop,cnz:BIT\_VECTOR(2 downto 0);

signal aq,ad,ab:BIT\_VECTOR(5 downto 0);

signal di,do,bo:BIT\_VECTOR(23 downto 0);

signal aret,areto:BIT\_VECTOR(20 downto 0);

signal maddr:natural;

type MICROINST is record -- ?????? ????????????

ACOP:bit\_vector(2 downto 0); -- ??? ???????? AU

AQ,AD,AB:bit\_vector(5 downto 0); -- ?????? FM

DI:BIT\_VECTOR(23 downto 0); -- ??????? ??????

START,WRD,RD:bit; -- ???? ??????????

end record;

constant n: positive:=13; --????? ???????????

type MICROPROGR is array(0 to n-1) of MICROINST;

----------microprogram----------

constant mp:MICROPROGR:=(

("100","000001","000000","000000",X"000000",'0','1','0'), --NOT 0000h, result = FFFFh in 000001

("101","000010","000001","000000",X"000000",'0','1','0'), --ABS FFFFh, result = 7FFFh in 000010

("110","000011","000010","000000",X"000000",'0','1','0'), --LSR 7FFFh, result = 3FFFh in 000011

("110","000011","000011","000000",X"000000",'0','1','0'), --LSR 3FFFh, result = 1FFFh in 000011

("001","000100","000011","000001",X"000000",'1','0','0'), --AND 3FFFh, 1FFFh, result = 1FFFh in 000100

("011","000101","000010","000100",X"000000",'1','0','0'), --SUB 7FFFh, 1FFFh, result = 6000h in 000101

("110","000110","000101","000000",X"000000",'0','1','0'), --LSR 6000h, result = 3000h in 000110

("111","001000","000001","000110",X"000000",'1','0','0'), --MUL FFFFh, 3000h, result = 97FFh in 001000 and D000h in 001001

("000","000111","000110","000101",X"000000",'1','0','0'), --ADD 3000h, 6000h, result = 9000h in 000111

("111","001010","000010","000110",X"000000",'1','0','0'), --MUL 7FFFh, 3000h, result = 17FFh in 001010 and D000h in 001011

("000","001110","001000","001010",X"000000",'1','0','0'), --ADD 97FFh, 17FFh, result = 0000h in 001110

("011","001111","001011","001001",X"000000",'1','0','0'), --SUB D000h, D000h, result = 0000h in 001111

("001","000000","001111","001110",X"000000",'1','0','0') --AND 0000h, 0000h, result = 0000h in out bus

);

begin

SYNCRO\_GEN: C<=not C after 5 ns; -- ????????? ?????????????

RESET\_GEN: RST<='1' ,'0' after 25 ns; -- ????????? c?????? ??????

CTM:process(C,RST)

begin -- ??????? ???????????

if RST='1' then

maddr<=0;

elsif C='1' and C'event then

if (RDY='1' and START='1') or WRD='1'or RD='1' then

maddr<=(maddr+1) mod n; -- +1 ? ????????

end if;

end if;

end process;

ROM\_U:(ACOP,AQ,AD,AB,DI,START,WRD,RD)<=mp(maddr);

AU\_U : AU port map (C,RST, --??????????? ?U

START => START,

RD => RD, WRD => WRD,

RET => RET, CALL => CALL,

DI => DI,

AB => AB, AD => AD, AQ => AQ,

ARET => ARET, ACOP => ACOP,

RDY => RDY, ARETO => ARETO,

DO => DO, BO=>BO, CNZ => CNZ);

end TB\_ARCHITECTURE;

**Результати симуляції:**