|  |  |
| --- | --- |
| **TECHNICKÁ UNIVERZITA V KOŠICIACH**  **FAKULTA ELEKTROTECHNIKY A INFORMATIKY** | |
| Obvod na realizáciu násobenia dvoch komplexných čísel v pevnej rádovej čiarke | |
| **2012** | **Ing. Norbert ÁDÁM, PhD.** |

**Obsah**

[Zoznam obrázkov 3](#_Toc341963941)

[Zoznam tabuliek 4](#_Toc341963942)

[Zoznam symbolov a skratiek 5](#_Toc341963943)

[Formulácia úlohy 6](#_Toc341963944)

[1 Analýza riešenia 7](#_Toc341963945)

[1.1 Voľba návrhového prostriedku a typu realizačného obvodu 7](#_Toc341963946)

[2 Návrh obvodu v jazyku VHDL 9](#_Toc341963947)

[2.1 Definovanie vstupných a výstupných portov obvodu 9](#_Toc341963948)

[2.2 Návrh operačnej časti obvodu 10](#_Toc341963949)

[2.3 Návrh riadiacej časti obvodu 11](#_Toc341963950)

[2.3.1 Kroky algoritmu výpočtu 12](#_Toc341963951)

[3 Syntéza obvodu 15](#_Toc341963952)

[4 Simulácia činnosti obvodu 17](#_Toc341963953)

[5 Návrh na urýchlenie výpočtu 19](#_Toc341963954)

[6 Záver 20](#_Toc341963955)

[Zoznam použitej literatúry 21](#_Toc341963956)

[Prílohy 22](#_Toc341963957)

Zoznam obrázkov

Obr. 1 Prepojenie funkčných stupňov 7

Obr. 2 Nastavenia 8

Obr. 3 Deklarácia knižníc a balíčkov 9

Obr. 4 Blok *entity* 10

Obr. 5 Blok *architecture* pre reprezentáciu toku dát operačnou časťou obvodu 11

Obr. 6 Stavový graf riadiaceho automatu 13

Obr. 7 Blok *architecture* pre reprezentáciu toku dát riadiacou časťou obvodu 14

Obr. 8 Výpis generovaný počas RTL syntézy nástrojom XST 15

Obr. 9 RTL schéme obvodu 16

Obr. 10 Stimulačný proces 17

Obr. 11 Overovací proces 17

Obr. 12 Simulačný výstup 18

Zoznam tabuliek

Tab. 1 Vstupné a výstupné porty obvodu 9

Tab. 2 Riadiace signály 12

Tab. 3 Využiteľnosť komponentov vývojovej dosky Spartan-6 SP605 Evaluation Platform 15

Zoznam symbolov a skratiek

ASIC **A**pplication **S**pecific **I**ntegrated **C**ircuits;

CAD **C**omputer **A**ided **D**esign; počítačom podporovaný návrh

FPGA **F**ield **P**rogrammable **G**ate **A**rrays

ISim nástroj slúžiaci na simuláciu činnosti logického obvodu; produkt firmy Xilinx

ns **n**ano **s**ekunda, 10-9

RA **r**iadiaci **a**utomat

XST nástroj slúžiaci na syntézu logického obvodu; produkt firmy Xilinx

Formulácia úlohy

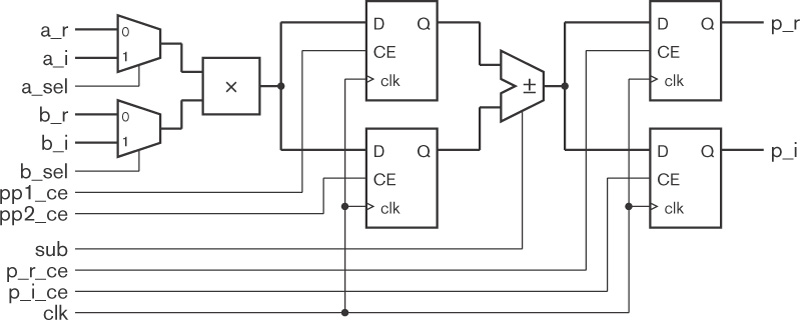
Navrhnite logický obvod na realizáciu operácie násobenia dvoch komplexných čísel. Reálna a imaginárna časť vstupných operandov je reprezentovaná v pevnej rádovej čiarke vo formáte : 4 bity pre vyjadrenie celej časti a 12 bitov na reprezentáciu desatinnej časti čísla. Na vyjadrenie celej časti výsledku výpočtu použite 8 bitov a 24 bitov pre desatinnú časť.

1. Analýza riešenia

Nech sú dané dve komplexné čísla a . Ich súčin sa vyjadruje v tvare:

(1)

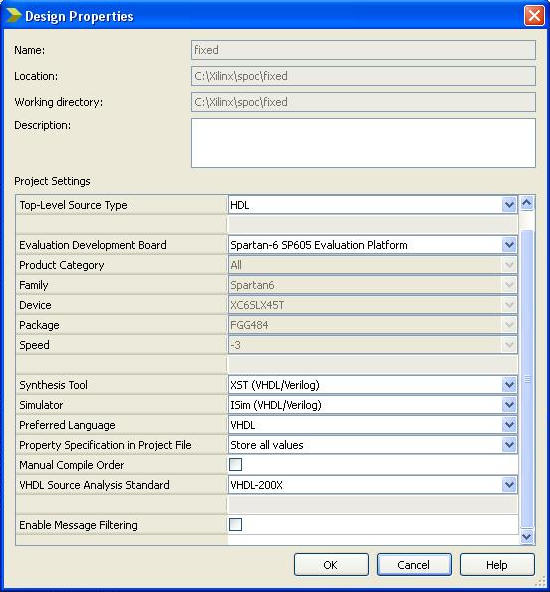
Zo vzťahu (1) vyplýva, že na realizáciu výpočtu potrebujeme vykonať 4 násobenia, jedno odčítanie a jedno sčítanie nad číslami v pevnej rádovej čiarke. V prípade, že by sme sa rozhodli riešiť úlohu na báze predchádzajúcej úvahy náš obvod by zaberal veľa miesta na čipe. Kvôli optimalizácii veľkosti plochy čipu namiesto štyroch násobičiek, jednej odčítačky a jednej sčítačky použijeme jednu násobičku a jednu kombinovanú sčítačku-odčítačku. Prepojenie funkčných stupňov (násobička, sčítačka-odčítačka, pomocné registre, multiplexory) je znázornené na Obr. 1.



Obr. 1 Prepojenie funkčných stupňov

* 1. Voľba návrhového prostriedku a typu realizačného obvodu

Návrhovým prostriedkom pre tvorbu funkčnej jednotky sa stal CAD systém od firmy XILINX s názvom ISE Design Suits 2012.3 v edícii ISE WebPACK [3]. Vzhľadom na skutočnosť, že operácie majú byť realizované v pevnej rádovej čiarke zvolila sa vývojová doska Spartan-6 SP605 Evaluation Platform (Obr. 2), ktorá natívne podporuje realizáciu výpočtov v pevnej rádovej čiarke [2]. Na syntézu obvodu sa použije nástroj XST [5], [7] a na simuláciu ISim [2], [4]. Opisným jazykom návrhu sa zvolil jazyk VHDL [5] so zohľadnením štandardu VHDL-200X.



Obr. 2 Nastavenia

1. Návrh obvodu v jazyku VHDL

Na prácu s pevnou rádovou čiarkou sa použila knižnica IEEE\_PROPOSED (Obr. 3). Použitie tejto knižnice je špecifické pre prostredie XILINX ISE Design Suits 2012.3 [2]. V inom prostredí (napr. Mentor HDL Designer 2012.1) sa použije balík IEEE.FIXED\_PKG z knižnice IEEE.

---------------------------------------------------------------------------------------------

-- Deklarácia knižníc a balíkov pre prostredie XILINX ISE Design Suits 2012.3

library IEEE, IEEE\_PROPOSED;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE\_PROPOSED.FIXED\_FLOAT\_TYPES.ALL;

use IEEE\_PROPOSED.FIXED\_PKG.ALL;

---------------------------------------------------------------------------------------------

-- Deklarácia knižníc a balíkov pre prostredie Mentor HDL Designer 2012.1

-- library IEEE;

-- use IEEE.STD\_LOGIC\_1164.ALL;

-- use IEEE.FIXED\_PKG.ALL;

---------------------------------------------------------------------------------------------

Obr. 3 Deklarácia knižníc a balíčkov

* 1. Definovanie vstupných a výstupných portov obvodu

Návrh začína špecifikáciou logického obvodu z pohľadu počtu a typu vstupných a výstupných portov (Tab. 1), ktorý je uvedený v deklaračnej časti bloku *entity* (Obr. 4).

Tab. 1 Vstupné a výstupné porty obvodu

|  |  |  |
| --- | --- | --- |
| Názov | Smer | Význam |
| clk | IN | Hodinový signál, ktorý sa použije na synchronizáciu činnosti jednotlivých komponentov obvodu. |
| reset | IN | Resetovací signál, ktorý sa použije na resetovanie pamäťovej časti obvodu. |
| input\_rdy | IN | Slúži na označenie dostupnosti operandov (a\_r, b\_r, a\_i, b\_i) na vstupe obvodu. |
| a\_r | IN | Reálna časť komplexného čísla A. |
| a\_i | IN | Imaginárna časť komplexného čísla A. |
| b\_r | IN | Reálna časť komplexného čísla B. |
| b\_i | IN | Imaginárna časť komplexného čísla B. |
| p\_r | OUT | Reálna časť výsledku operácie P = A\*B. |
| p\_i | OUT | Imaginárna časť výsledku operácie P = A\*B. |

entity multiplier is

Port ( clk : in STD\_LOGIC;

reset : in STD\_LOGIC;

input\_rdy : in STD\_LOGIC;

a\_r, a\_i : in sfixed(3 downto -12);

b\_r, b\_i : in sfixed(3 downto -12);

p\_r, p\_i : out sfixed(7 downto -24));

end multiplier;

Obr. 4 Blok *entity*

* 1. Návrh operačnej časti obvodu

Navrhnutý obvod (Obr. 1) pozostáva z riadiacej (na obrázku vyznačená nie je) a z operačnej časti. Opis toku dát v jazyku VHDL operačnou časťou navrhnutého logického obvodu je znázornený na obrázku Obr. 5.

architecture rtl of multiplier is

signal a\_sel, b\_sel,

pp1\_ce, pp2\_ce,

sub, p\_r\_ce, p\_i\_ce : std\_logic;

signal a\_operand, b\_operand : sfixed(3 downto -12);

signal pp, pp1, pp2, sum : sfixed(7 downto -24);

signal plus, minus : sfixed(8 downto -24);

...

begin

a\_operand <= a\_r when a\_sel = '0' else a\_i;

b\_operand <= b\_r when b\_sel = '0' else b\_i;

pp <= a\_operand \* b\_operand;

pp1\_reg : process (clk) is

begin

if rising\_edge(clk) then

if pp1\_ce = '1' then

pp1 <= pp;

end if;

end if;

end process pp1\_reg;

pp2\_reg : process (clk) is

begin

if rising\_edge(clk) then

if pp2\_ce = '1' then

pp2 <= pp;

end if;

end if;

end process pp2\_reg;

plus <= pp1 + pp2;

minus <= pp1 - pp2;

sum <= plus(7 downto -24) when sub = '0' else minus(7 downto -24);

p\_r\_reg : process (clk) is

begin

if rising\_edge(clk) then

if p\_r\_ce = '1' then

p\_r <= sum;

end if;

end if;

end process p\_r\_reg;

p\_i\_reg : process (clk) is

begin

if rising\_edge(clk) then

if p\_i\_ce = '1' then

p\_i <= sum;

end if;

end if;

end process p\_i\_reg;

...

end architecture rtl;

Obr. 5 Blok *architecture* pre reprezentáciu toku dát operačnou časťou obvodu

Signály uvedené v deklaračnej časti bloku *architecture* reprezentujú prepojenia medzi komponentmi obvodu a riadiace signály generované riadiacou časťou obvodu (Obr. 7).

Multiplexory na vstupe obvodu (Obr. 1) sú reprezentované zápisom do signálov *a\_operand* a *b\_operand* na základe hodnoty riadiacich signálov *a\_sel* a *b\_sel*, ktoré fungujú ako selektory pre reálnu a imaginárnu časť vstupných komplexných čísel *a* a *b*.

Zápis do pomocného signálu *pp* rieši operáciu násobenia príslušných zložiek komplexných čísel podľa vzťahu (1). Procesy *pp1\_reg* a *pp2\_reg* fungujú ako záchytné registre pre čiastkové súčiny

(2)

Výstup z kombinovanej sčítačky-odčítačky je prezentovaný signálom *sum*.

* 1. Návrh riadiacej časti obvodu

Návrh riadiacej časti obvodu je založený na určení postupnosti krokov na realizáciu výpočtu podľa vzťahu (1). Existuje niekoľko možností v akom poradí je možné vykonať kroky výpočtu. V nasledujúcej časti dokumentu sa uvedie jedna z možností.

* + 1. Kroky algoritmu výpočtu

Kvôli zjednodušeniu návrhu, povolí sa aktivácia v danom hodinovom takte (*clk*) len jedného funkčného stupňa obvodu. Výpočet je odštartovaný nastavením signálu *input\_*rdy = ‘1’.

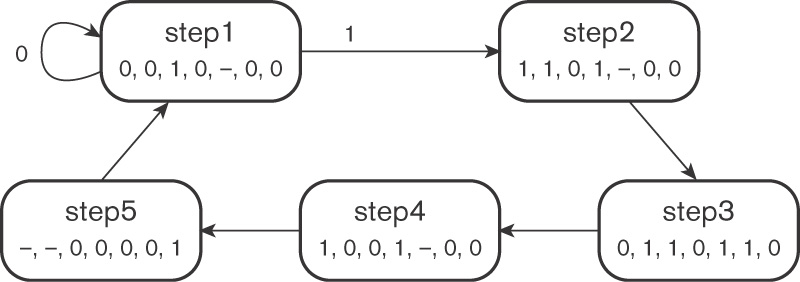
|  |  |
| --- | --- |
| **Krok.1** | Vynásob operandy *a\_r* a *b\_r*. Ulož výsledok súčinu do registra 1. |
| **Krok.2** | Vynásob operandy *a\_i* a *b\_i*. Ulož výsledok súčinu do registra 2. |
| **Krok.3** | Odčítaj od hodnoty v registri 1 hodnotu v registri 2. Ulož výsledok výpočtu do registra pre uchovanie reálnej časti výsledku P. |
| **Krok.4** | Vynásob operandy *a\_r* a *b\_i*. Ulož výsledok súčinu do registra 1. |
| **Krok.5** | Vynásob operandy *a\_i* a *b\_r*. Ulož výsledok súčinu do registra 2. |
| **Krok.6** | Pripočítaj k hodnote z registra 1 hodnotu z registra 2. Ulož výsledok výpočtu do registra pre uchovanie imaginárnej časti výsledku P. |

Riadiaci automat (RA) sa navrhne na báze automatu Moore. Zlúčením krokov 3 a 4 (medzi krokmi 3 a 4 sa neprejaví zdrojový hazard), počet stavov RA sa rovná číslu päť. Počet potrebných riadiacich signálov, ich názvy ako aj informácia o tom, v ktorom hodinovom takte (*clk*), ktorý signál má byť aktívny sa uvádza v Tab. 2.

Tab. 2 Riadiace signály

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| stav/krok | a\_sel | b\_sel | pp1\_ce | pp2\_ce | sub | p\_r\_ce | p\_i\_ce |
| 1 | 0 | 0 | 1 | 0 | -- | 0 | 0 |
| 2 | 1 | 1 | 0 | 1 | -- | 0 | 0 |
| 3 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 4 | 1 | 0 | 0 | 1 | -- | 0 | 0 |
| 5 | -- | -- | 0 | 0 | 0 | 0 | 1 |

Potom stavový graf (Obr. 6) RA typu Moore má tvar



Obr. 6 Stavový graf riadiaceho automatu

Opis riadiacej časti obvodu je znázornený na Obr. 7.

architecture rtl of multiplier is

signal a\_sel, b\_sel,

pp1\_ce, pp2\_ce,

sub, p\_r\_ce, p\_i\_ce : std\_logic;

signal a\_operand, b\_operand : sfixed(3 downto -12);

signal pp, pp1, pp2, sum : sfixed(7 downto -24);

signal plus, minus : sfixed(8 downto -24);

type multiplier\_state is (step1, step2, step3, step4, step5);

signal current\_state, next\_state : multiplier\_state;

begin

...

state\_reg : process (clk, reset) is

begin

if reset = '1' then

current\_state <= step1;

elsif rising\_edge(clk) then

current\_state <= next\_state;

end if;

end process state\_reg;

next\_state\_logic : process (current\_state, input\_rdy) is

begin

case current\_state is

when step1 =>

if input\_rdy = '0' then

next\_state <= step1;

else

next\_state <= step2;

end if;

when step2 =>

next\_state <= step3;

when step3 =>

next\_state <= step4;

when step4 =>

next\_state <= step5;

when step5 =>

next\_state <= step1;

end case;

end process next\_state\_logic;

output\_logic : process (current\_state) is

begin

a\_sel <= '0'; b\_sel <= '0'; pp1\_ce <= '0'; pp2\_ce <= '0';

sub <= '0'; p\_r\_ce <= '0'; p\_i\_ce <= '0';

case current\_state is

when step1 =>

pp1\_ce <= '1';

when step2 =>

a\_sel <= '1'; b\_sel <= '1'; pp2\_ce <= '1';

when step3 =>

b\_sel <= '1'; pp1\_ce <= '1'; sub <= '1'; p\_r\_ce <= '1';

when step4 =>

a\_sel <= '1'; pp2\_ce <= '1';

when step5 =>

p\_i\_ce <= '1';

end case;

end process output\_logic;

end architecture rtl;

Obr. 7 Blok *architecture* pre reprezentáciu toku dát riadiacou časťou obvodu

1. Syntéza obvodu

Nasledujúci výpis generovaný počas RTL syntézy obvodu nám poskytuje informáciu o kódovaní stavov stavového automatu a o časovaní signálov.

=============================================================

\* Low Level Synthesis \*

=============================================================

Analyzing FSM <MFsm> for best encoding.

Optimizing FSM <FSM\_0> on signal <current\_state[1:5]> with one-hot encoding.

-------------------

State | Encoding

-------------------

step1 | 00001

step2 | 00010

step3 | 00100

step4 | 01000

step5 | 10000

-------------------

Timing Summary:

---------------

Speed Grade: -3

Minimum period: 6.468ns (Maximum Frequency: 154.598MHz)

Minimum input arrival time before clock: 6.984ns

Maximum output required time after clock: 3.597ns

Maximum combinational path delay: No path found

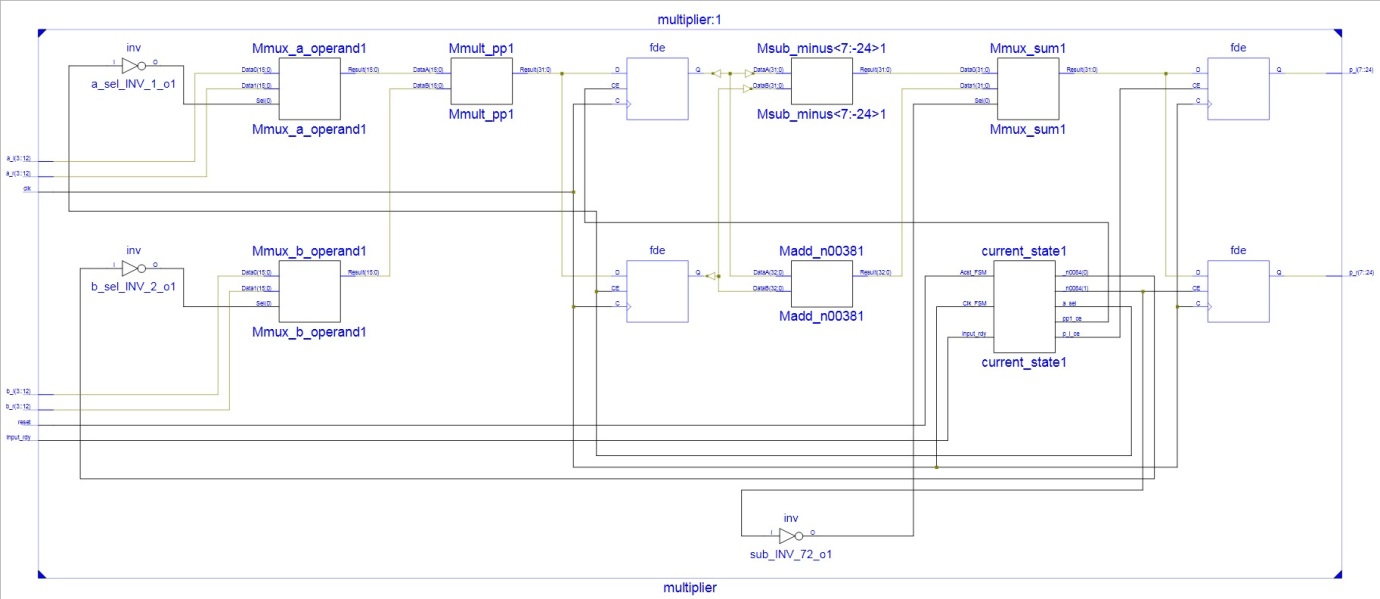
Obr. 8 Výpis generovaný počas RTL syntézy nástrojom XST

Využiteľnosť komponentov dosky je znázornená v Tab. 3.

Tab. 3 Využiteľnosť komponentov vývojovej dosky Spartan-6 SP605 Evaluation Platform

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Device Utilization Summary (estimated values)** | | | | **[-]** |
| **Logic Utilization** | **Used** | **Available** | **Utilization** | |
| Number of Slice Registers | 135 | 54576 | 0% | |
| Number of Slice LUTs | 132 | 27288 | 0% | |
| Number of fully used LUT-FF pairs | 51 | 216 | 23% | |
| Number of bonded IOBs | 131 | 296 | 44% | |
| Number of BUFG/BUFGCTRLs | 1 | 16 | 6% | |
| Number of DSP48A1s | 1 | 58 | 1% | |

RTL schéma zapojenia obvodu je znázornená na Obr. 9.



Obr. 9 RTL schéme obvodu

1. Simulácia činnosti obvodu

Na simuláciu činnosti obvodu sa použil nástroj ISim, ktorý však nie je schopný znázorniť čísla vo formáte pevnej rádovej čiarky (sfixed), preto sa použila hexadecimálna reprezentácia pre operandy *a\_r*, *b\_r*, *a\_i*, *b\_i*, *p\_r* a *p\_i* (Obr. 12).

Na overenie korektnosti návrhu sa použila simulačná zostava pozostávajúca zo stimulačného procesu (Obr. 10)

...

reset <= '1', '0' after 2 \* clk\_period;

apply\_test\_cases : process is

procedure apply\_test ( a\_test, b\_test : in complex ) is

begin

a <= a\_test; b <= b\_test; input\_rdy <= '1';

wait until falling\_edge(clk); input\_rdy <= '0';

for i in 1 to 5 loop

wait until falling\_edge(clk);

end loop;

end procedure apply\_test;

begin

wait until falling\_edge(clk) and reset = '0';

apply\_test(cmplx(0.0, 0.0), cmplx(1.0, 2.0));

apply\_test(cmplx(1.0, 1.0), cmplx(1.0, 1.0));

-- further test cases ...

wait;

end process apply\_test\_cases;

a\_r <= to\_sfixed(a.re, a\_r'left, a\_r'right);

a\_i <= to\_sfixed(a.im, a\_i'left, a\_i'right);

b\_r <= to\_sfixed(b.re, b\_r'left, b\_r'right);

b\_i <= to\_sfixed(b.im, b\_i'left, b\_i'right);

...

Obr. 10 Stimulačný proces

a z overovacieho procesu (Obr. 11).

...

check\_outputs : process is

variable p : complex;

begin

wait until rising\_edge(clk) and input\_rdy = '1';

p := a \* b;

for i in 1 to 5 loop

wait until falling\_edge(clk);

end loop;

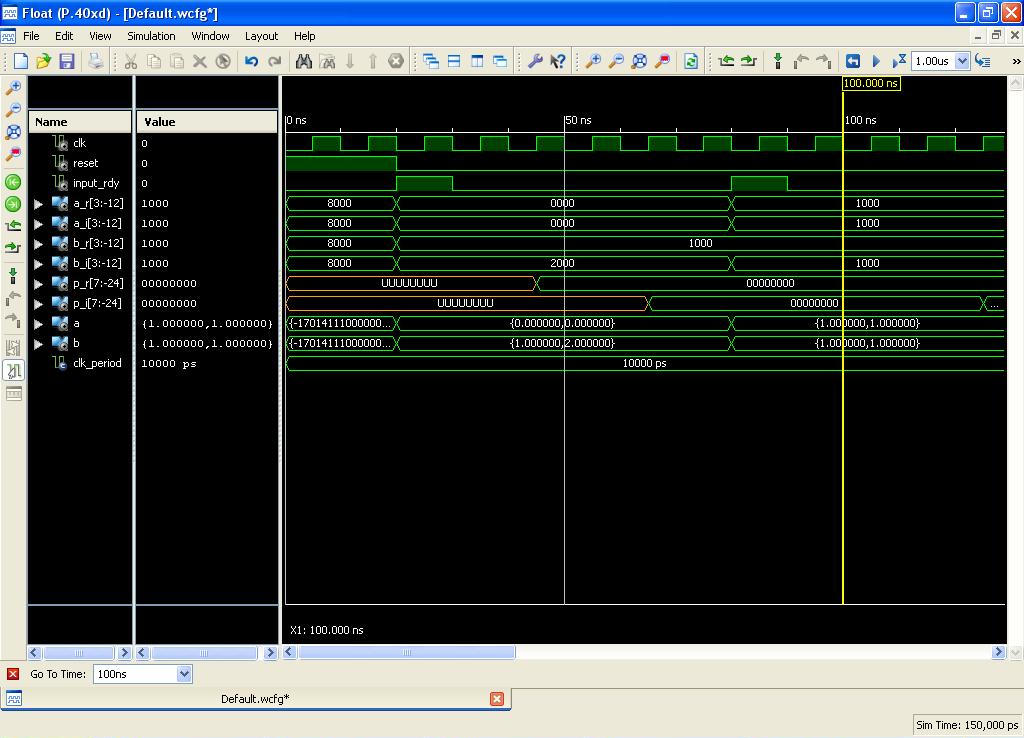
assert abs (to\_real(p\_r) - p.re) < 2.0\*\*(-12) and abs (to\_real(p\_i) - p.im) < 2.0\*\*(-12);

end process check\_outputs;

...

Obr. 11 Overovací proces

Na základe vykonanej simulácie, simulačné priebehy (Obr. 12) dokazujú, že obvod pracuje správne.



Obr. 12 Simulačný výstup

1. Návrh na urýchlenie výpočtu

V prípade, že sa použijú dve násobičky, výpočet reálnej a imaginárnej časti výsledku násobenia dvoch komplexných čísel môže byť realizovaný v rovnakom hodinovom takte (*clk*). Tento predpoklad nám umožňuje zrealizovať výpočet za tri kroky :

|  |  |
| --- | --- |
| **Krok.1** | Vynásob a\_r a b\_r a ulož výsledok do registra 1; vynásob a\_i a b\_i a ulož výsledok do registra 2; |
| **Krok.2** | Odčítaj od obsahu regista 1 obsah registra 2 a ulož výsledok výpočtu do registra pre reálnu časť výsledku P; vynásob a\_r a b\_i a ulož výsledok do registra 1; vynásob a\_i a b\_r a ulož výsledok do registra 2; |
| **Krok.3** | Pripočítaj k obsahu regista 1 obsah registra 2 a ulož výsledok výpočtu do registra pre imaginárnu časť výsledku P; |

1. Záver

Úlohou tejto práce bolo navrhnúť logický obvod na realizáciu operácie násobenia dvoch komplexných čísel v pevnej rádovej čiarke. Navrhnutý obvod dokáže pracovať na frekvencií 154.598MHz. ....

Zoznam použitej literatúry

1. ASHENDEN, Peter J. Digital Design. An Embedded System Approach Using VHDL, 2008. 600 s. ISBN 978-0-12-369528-4.
2. ISE Simulator (ISim). Dostupné na internete :

http://www.xilinx.com/tools/isim.htm

1. ISE WebPACK Design Software. Dostupné na internete :

http://www.xilinx.com/products/design-tools/ise-design-suite/ise-webpack.htm

1. ISim User Guide. Dostupné na internete :

http://www.xilinx.com/support/documentation/sw\_manuals/xilinx11/plugin\_ism.pdf

1. PINKER, J – POUPA, M. Číslicové systémy a jazyk VHDL, 2006. 352 s. ISBN 80-7300-198-5
2. XST Synthesis Overview. Dostupné na internete:

http://www.xilinx.com/support/documentation/sw\_manuals/xilinx11/ise\_c\_using\_xst\_for\_synthesis.htm

1. XST User Guide. Dostupné na internete :

http://www.xilinx.com/itp/xilinx10/books/docs/xst/xst.pdf

1. XST User Guide for Virtex-6 and Spartan-6 Devices. Dostupné na internete :

http://www.xilinx.com/support/documentation/sw\_manuals/xilinx12\_4/xst\_v6s6.pdf

Prílohy

1. Súbor multiplier.vhd.
2. Súbor multiplier\_tb.vhd
3. RTL schéme vo formáte XST.