|  |  |  |  |
| --- | --- | --- | --- |
| **VHDL** | | | |
| **Andrzej Kmicic** | **23 XI 2018** | **Wt 12:45** | **G2** |

Spis treści

[1. Zajęcia nr 1 1](#__RefHeading___Toc320_463844247)

[1.1. „Hello World” 1](#__RefHeading___Toc322_463844247)

[1.2. Migotanie diodą LED z wykorzystaniem pętli opóźniających (przykład) 1](#__RefHeading___Toc330_463844247)

[1.3. Kolejny program (przykład) 2](#__RefHeading___Toc338_463844247)

[2. Zajęcia nr 2 3](#__RefHeading___Toc346_463844247)

[2.1. Kolejny program (przykład) 3](#__RefHeading___Toc348_463844247)

# Zajęcia nr 1

## Hello World

### Opis działania programu

Program typu „Hello World” w języku VHDL. Przepisanie wartości z przełączników na pakiet LED.

### Kod źródłowy (najważniejsze fragmenty)

entity hello\_world is

Port ( SW : inout STD\_LOGIC\_VECTOR (15 downto 0);

LED : out STD\_LOGIC\_VECTOR (15 downto 0));

end hello\_world;

architecture Behavioral of hello\_world is

begin

LWD <= SW;

end Behavioral;

### Wnioski

## Bramki logiczne

### Opis działania programu

Celem programu było wystawienie na wyjścia LED wyników operacji logicznych.

### Kod źródłowy (najważniejsze fragmenty)

entity gates is

Port ( SW : inout STD\_LOGIC\_VECTOR (15 downto 0);

LED : out STD\_LOGIC\_VECTOR (15 downto 0));

end gates;

architecture Behavioral of gates is

begin

LED (1) <= SW(0) and SW(1);

LED (2) <= not (SW(0) and SW(1));

LED (3) <= SW(0) or SW(1);

LED (4) <= not (SW(0) or SW(1));

LED (5) <= SW(0) xor SW(1);

LED (6) <= not (SW(0) xor SW(1));

LED (7) <= not SW(0);

end Behavioral;

### Wnioski

Podstawowe działa na bitach są wykonywane przy pomocy operatorów zgodnych z nazwą wykonywanej operacji.

## Bramki logiczne z centralnym wyłącznikiem

### Opis działania programu

Celem programu było zaimplementowanie programu realizującego bramki logiczne z wyróżnionym przełącznikiem, którego niski stan wyłączał wszystkie diody LED. Funkcjonalność ta została osiągnięta poprzez stworzenie tymczasowego wektora bitów, który reprezentował wyjście. W przypadku wysokiego stanu przełącznika centralnego, wektor tymczasowy był przepisywany na wyjście, w przeciwnym przypadku wyjścia zostawały zerowane.

### Kod źródłowy (najważniejsze fragmenty)

### entity gates\_with\_central\_switch is

### Port ( SW : inout STD\_LOGIC\_VECTOR (15 downto 0);

### LED : out STD\_LOGIC\_VECTOR (15 downto 0));

### end gates\_with\_central\_switch;

### architecture Behavioral of gates\_with\_central\_switch is

### signal TMP\_LED: STD\_LOGIC\_VECTOR (15 downto 0);

### begin

### LED <= TMP\_LED when SW(15)='1' else

### "0000000000000000";

### TMP\_LED (1) <= SW(0) and SW(1);

### TMP\_LED (2) <= not (SW(0) and SW(1));

### TMP\_LED (3) <= SW(0) or SW(1);

### TMP\_LED (4) <= not (SW(0) or SW(1));

### TMP\_LED (5) <= SW(0) xor SW(1);

### TMP\_LED (6) <= not (SW(0) xor SW(1));

### TMP\_LED (7) <= not SW(0);

### end Behavioral;Wnioski

Zaimplementowany pogram prawidłowo migotał diod ami LED. Z niewykorzystaniem pojedynczych pętli opóźniających nie można uzyskać częstotliwości migotania mniejszej niż 5Hz.

## ****Bramki logiczne z centralnym wyłącznikiem i zamianą kolejności bitów****.

### Opis działania programu

Celem programu było wystawienie wyjścia z tymczasowego wektora na wyjścia LED z zamienioną kolejnością bitów.

### Kod źródłowy (najważniejsze fragmenty)

entity gates\_bit\_mix is

Port ( SW : inout STD\_LOGIC\_VECTOR (15 downto 0);

LED : out STD\_LOGIC\_VECTOR (15 downto 0));

end gates\_bit\_mix;

architecture Behavioral of gates\_bit\_mix is

signal TMP\_LED: STD\_LOGIC\_VECTOR (15 downto 0);

begin

LED (7 downto 1) <= TMP\_LED(4 downto 2) & TMP\_LED(5) & TMP\_LED(6) & TMP\_LED(1 downto 0) when SW(15)='1' else

"0000000";

TMP\_LED (0) <= SW(0) and SW(1);

TMP\_LED (1) <= not (SW(0) and SW(1));

TMP\_LED (6) <= SW(0) or SW(1);

TMP\_LED (5) <= not (SW(0) or SW(1));

TMP\_LED (2) <= SW(0) xor SW(1);

TMP\_LED (3) <= not (SW(0) xor SW(1));

TMP\_LED (4) <= not SW(0);

end Behavioral;

# Zajęcia nr 2

## Kolejny program (przykład)

### Opis działania programu

Celem programu było migotanie jedną z diod LED podłączonych do mikrokontrolera. Częstotliwość migotania nie musiała być precyzyjnie ustawiona.

### Kod źródłowy (najważniejsze fragmenty)

org 0x00

JMP MAIN;

org 0x30

MAIN:

MOV P1,#0x00; Zaświecenie 8 diod LED

CALL WAIT; Wywołanie pętli opóźniającej trwającej około 120ms

MOV P1,#0xFF; Zgaszenie 8 diod LED

CALL WAIT; Wywołanie pętli opóźniającej trwającej około 120ms

JMP MAIN; Zapętlenie nieskończone

### Wnioski

Zaimplementowany pogram prawidłowo migotał diod ami LED. Z niewykorzystaniem pojedynczych pętli opóźniających nie można uzyskać częstotliwości migotania mniejszej niż 5Hz.