



**Języki Opisu Sprzętu**  
Projekt: Elektroniczny Sejf Hotelowy  
Dokumentacja

Arkadiusz Kasprzak  
Jarosław Cierpich  
Wydział Fizyki i Informatyki Stosowanej  
Informatyka Stosowana

30 listopada 2019

# Spis treści

<b>1</b>	<b>Wstęp</b>	<b>3</b>
<b>2</b>	<b>Projekt</b>	<b>3</b>
2.1	Założenia projektowe . . . . .	3
2.2	Wymagana funkcjonalność . . . . .	3
<b>3</b>	<b>Dokumentacja użytkownika</b>	<b>4</b>
<b>4</b>	<b>Dokumentacja techniczna</b>	<b>4</b>
4.1	Warstwa Hardware . . . . .	4
4.2	Warstwa Software - architektura . . . . .	4
4.3	Warstwa Software - parametry i moduły projektu . . . . .	4
4.3.1	Parametry projektu . . . . .	4
4.3.2	Moduły projektu . . . . .	4
<b>5</b>	<b>Analiza procesu syntezy</b>	<b>5</b>
<b>6</b>	<b>Testy</b>	<b>5</b>
6.1	Moduły testujące . . . . .	5
6.1.1	Testy modułu Bcd2Dec . . . . .	7
6.1.2	Testy modułu ClkDiv . . . . .	7
6.2	Testy manualne . . . . .	7

# 1 Wstęp

Niniejszy dokument stanowi dokumentację projektu **Elektroniczny Sejf Hotelowy** wykonanego w ramach przedmiotu **Języki Opisu Sprzętu** (WFIS AGH) przez Jarosława Cierpicha i Arkadiusza Kasprzaka. Dokument ten zawiera m.in. założenia projektowe oraz opis wymaganej funkcjonalności, dokumentację przeznaczoną dla użytkownika projektu, dokumentację techniczną, analizę procesu syntezy oraz opis procedury testowania.

## 2 Projekt

Ten rozdział poświęcony został opisowi założeń projektowych oraz wymaganej funkcjonalności.

### 2.1 Założenia projektowe

Projekt dostarczać ma funkcji elektronicznego sejfu hotelowego, to znaczy pozwalać ma na otwieranie go za pomocą z góry ustalonego szyfru oraz późniejsze zamknięcie go. Projekt ma być zrealizowany na płycie rozwojowej **Zedboard** (do Xilinx Zynq-7000). Ze względu na okoliczności wykonywania projektu (projekt jako zaliczenie przedmiotu) przyjęte zostały pewne uproszczenia:

- czujnik zamknięcia sejfu zastąpiony zostaje przełącznikiem obsługiwany przez użytkownika
- szyfr jest parametrem projektu - nie jest możliwa jego modyfikacja na płycie bez powtórzenia procesu syntezy i implementacji
- ruch rygla reprezentowany jest za pomocą dwóch diod LED dostępnych na płycie

Do realizacji projektu użyty ma być język **SystemVerilog** oraz środowisko **Xilinx Vivado**. Szyfr składać ma się z trzech liczb z przedziału [0; 32). Liczby mają być wprowadzane przez użytkownika za pomocą pokrętła, przy czym kierunek obrotu pokrętła wskazuje, która liczba jest aktualnie wprowadzana: ruch zgodny ze wskazówkami zegara oznacza pierwszą lub trzecią liczbę, ruch przeciwny do wskazówek zegara - drugą liczbę. Wprowadzenie nieprawidłowej liczby ma przerywać proces podawania szyfru - tzn. układ nie czeka, aż wprowadzony zostanie cały szyfr. Aktualnie wprowadzona liczba ma być prezentowana za pomocą wyświetlacza OLED. Układ ma być w miarę możliwości pozbawiony błędów związanych z drganiami styków czy niestandardowym działaniem użytkownika.

### 2.2 Wymagana funkcjonalność

Od projektu wymaga się dostarczenia następującej funkcjonalności:

- możliwość wprowadzenia przez użytkownika poprawnego szyfru składającego się z trzech liczb z zakresu [0; 32) za pomocą pokrętła
- możliwość wprowadzenia przez użytkownika niepoprawnego szyfru, co automatycznie przerwać ma proces otwierania sejfu
- możliwość monitorowania przez użytkownika aktualnego stanu otwarcia sejfu - za pomocą diod LED
- możliwość monitorowania przez użytkownika aktualnie wprowadzanej wartości - za pomocą wyświetlacza OLED
- możliwość rozpoczęcia procesu otwierania sejfu - za pomocą przycisku *Open*

- możliwość zamknięcia sejfu - za pomocą przycisku *Close*
- możliwość ustalenia aktualnej pozycji rygla sejfu - za pomocą przełącznika - zastępuje to czujnik zamknięcia sejfu
- możliwość łatwego zaprogramowania nowego szyfru - zmiana trzech wartości w kodzie

### 3 Dokumentacja użytkownika

### 4 Dokumentacja techniczna

#### 4.1 Warstwa Hardware

#### 4.2 Warstwa Software - architektura

#### 4.3 Warstwa Software - parametry i moduły projektu

Ten podrozdział opisuje część implementacji projektu - zastosowane parametry oraz moduły napisane w języku SystemVerilog.

##### 4.3.1 Parametry projektu

Główny moduł projektu udostępnia następujące **parametry** pozwalające na modyfikację działania sejfu:

- *slowClockPeriodLength* - współczynnik oznaczający wartość dzielnika częstotliwości zegara, którym taktowany jest sejf (z wyjątkiem debouncerów i wyświetlacza OLED). Wartość domyślna: 100000.
- *debouncerClockPeriodLength* - współczynnik oznaczający wartość dzielnika częstotliwości zegara, którym taktowane są debouncery w projekcie. Wartość domyślna: 300007.
- *areDebouncersUsed* - flaga włączająca lub wyłączająca generację debouncerów w projekcie. Wartość domyślna: 1 - debouncery mają zostać wygenerowane.
- *firstCodeNumber* - pierwsza liczba szyfru. Wartość domyślna: 15.
- *secondCodeNumber* - druga liczba szyfru. Wartość domyślna: 30.
- *thirdCodeNumber* - trzecia liczba szyfru. Wartość domyślna: 9.

##### 4.3.2 Moduły projektu

Projekt składa się z następujących **modułów**:

- SafeTop - główny moduł projektu
- Bcd2Decades - licznik BCD (Binary Coded Decimal) o dwóch dekadach
- ClockDiv - dzielnik zegara
- DebouncerButtons - układ wygaszający drgania na przyciskach
- DebouncerKnob - układ wygaszający drgania na pokrętle

- DigitCompare - układ porównujący wprowadzone przez użytkownika liczby z szyfrem
- LedDriver - sterownik diod LED
- MasterFsm - główna logika sejfu
- RseDecoder - układ odpowiedzialny za komunikację z pokrętle
- Moduły odpowiedzialne za obsługę wyświetlacza OLED:
  - OledDriver
  - delay
  - fsm\_init
  - fsm\_oper
  - rom
  - spi
  - update\_page

Ponadto projekt zawiera **dwa inne pliki** związane z obsługą wyświetlacza OLED:

- screens.vh
- pixel\_SSD1306.dat

W dalszej części zaprezentowane zostanie działanie poszczególnych modułów.

**SafeTop**

## 5 Analiza procesu syntezy

## 6 Testy

Ostatni rozdział poświęcony został procesowi testowania projektu - w tym przygotowanym modułom testowym oraz procesowi testowania manualnego.

### 6.1 Moduły testujące

Projekt zawiera **TUTAJ WSTAWIC ILE** modułów testowych (tzw. moduły *Testbench*). Umożliwiają one przeprowadzenie symulacji działania modułów projektu. Moduły testowane były za pomocą dwóch typów symulacji:

- symulacja behawioralna (*behavioral simulation*)
- symulacja po syntezie z uwzględnieniem parametrów czasowych (*post-synthesis timing simulation*)

**TUTAJ WSTAWIC STRUKTURE TB** Podstawowa struktura większości modułów *Testbench* jest podobna - składają się one z:

- deklaracji parametrów wejściowych modułu (jeśli takie są)

- deklaracji zmiennych stanowiących wejścia i wyjścia testowanego modułu oraz zmiennej odpowiadającej za *Global System Reset* - *GSR*
- instancji testowanego modułu (*UUT - Unit Under Test*)
- generacji sygnałów wejściowych testowanego modułu (w tym zwykle sygnału zegara i resetu)

Listing 1 ilustruje opisaną powyżej strukturę.

Listing 1: Uproszczona struktura wykonanych modułów testujących

```
module TbExample();

    // parametry
    localparam mod = 3;

    // wejścia
    reg clk, rst;
    reg in1;

    // wyjścia
    reg [3:0] out1;

    // ...

    // GSR - Global System Reset
    wire gsr = glbl.GSR;

    // UUT - Unit Under Test
    ExampleModule #(.mod(mod)) EXAMPLE (
        .clk(clk), .rst(rst), .in1(in1), .out1(out1));

    // generacja sygnałów wejściowych

    // zegar
    initial begin
        clk = 1'b0;
        @(negedge gsr);
        forever #5 clk = ~clk;
    end

    // reset
    initial begin
        rst = 1'b1;
        @(negedge gsr);
        #5 rst = 1'b0;
    end

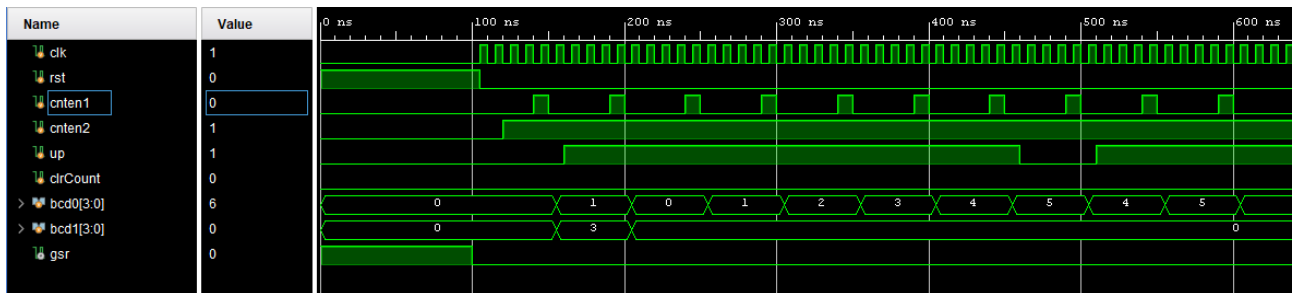
    // in1
    initial begin
        // kod generujący wartości sygnału in1
    end

    // ...
```

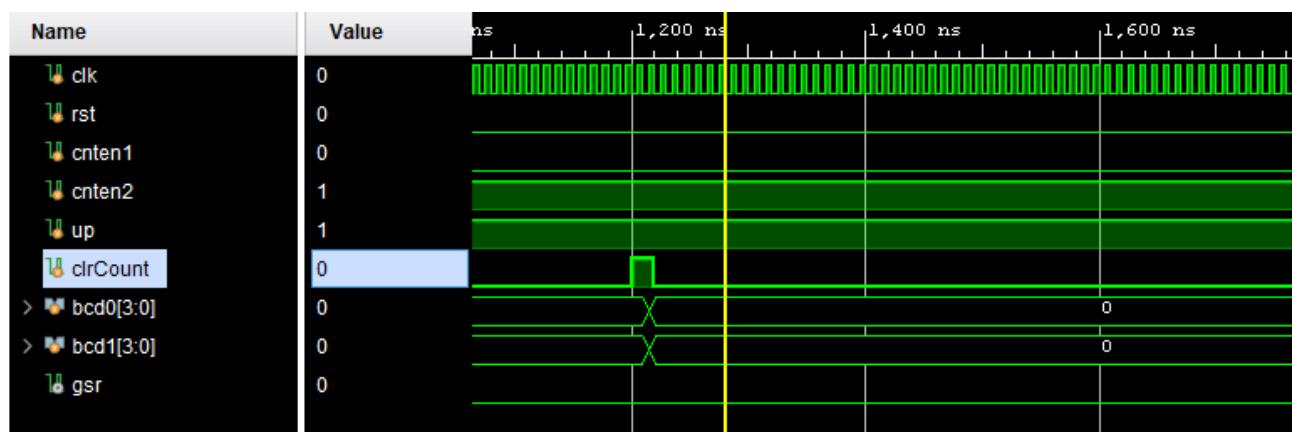
```
endmodule
```

W dalszej części tego podrozdziału omówione zostaną poszczególne moduły testujące oraz wyniki przeprowadzonych symulacji behawioralnych.

### 6.1.1 Testy modułu Bcd2Dec



Rysunek 1: Tutaj dac opis



Rysunek 2: Tutaj dac opis

### 6.1.2 Testy modułu ClkDiv

## 6.2 Testy manualne