



Języki Opisu Sprzętu
Projekt: Elektroniczny Sejf Hotelowy
Dokumentacja

Arkadiusz Kasprzak
Jarosław Cierpich
Wydział Fizyki i Informatyki Stosowanej
Informatyka Stosowana

30 listopada 2019

Spis treści

1	Wstęp	3
2	Projekt	3
2.1	Założenia projektowe	3
2.2	Wymagana funkcjonalność	3
3	Dokumentacja użytkownika	4
4	Dokumentacja techniczna	4
4.1	Warstwa Hardware	4
4.2	Warstwa Software - architektura	4
4.3	Warstwa Software - parametry i moduły projektu	6
4.3.1	Parametry projektu	6
4.3.2	Lista modułów projektu	7
4.3.3	Moduł SafeTop	8
4.3.4	Moduł RseDecoder	8
4.3.5	Moduł Bcd2Decades	8
4.3.6	Moduł MasterFsm	8
5	Analiza raportu syntezy	8
5.1	Kodowanie stanów	8
5.2	Użyte zasoby	8
5.3	Parametry czasowe	8
6	Testy	8
6.1	Moduły testujące	9
6.1.1	Testy modułu SafeTop	10
6.1.2	Testy modułu Bcd2Decades	10
6.1.3	Testy modułu RseDecoder	11
6.1.4	Testy modułu MasterFsm	11
6.1.5	Testy modułu ClkDiv	11
6.1.6	Testy modułu DebouncerButtons	11
6.1.7	Testy modułu DigitCompare	11
6.1.8	Testy modułu OledDriver	11
6.1.9	Testy modułu Delay	11
6.2	Testy manualne	11
7	Możliwe udoskonalenia	11

1 Wstęp

Niniejszy dokument stanowi dokumentację projektu **Elektroniczny Sejf Hotelowy** wykonanego w ramach przedmiotu **Języki Opisu Sprzętu** (WFiIS 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 drganiem 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*

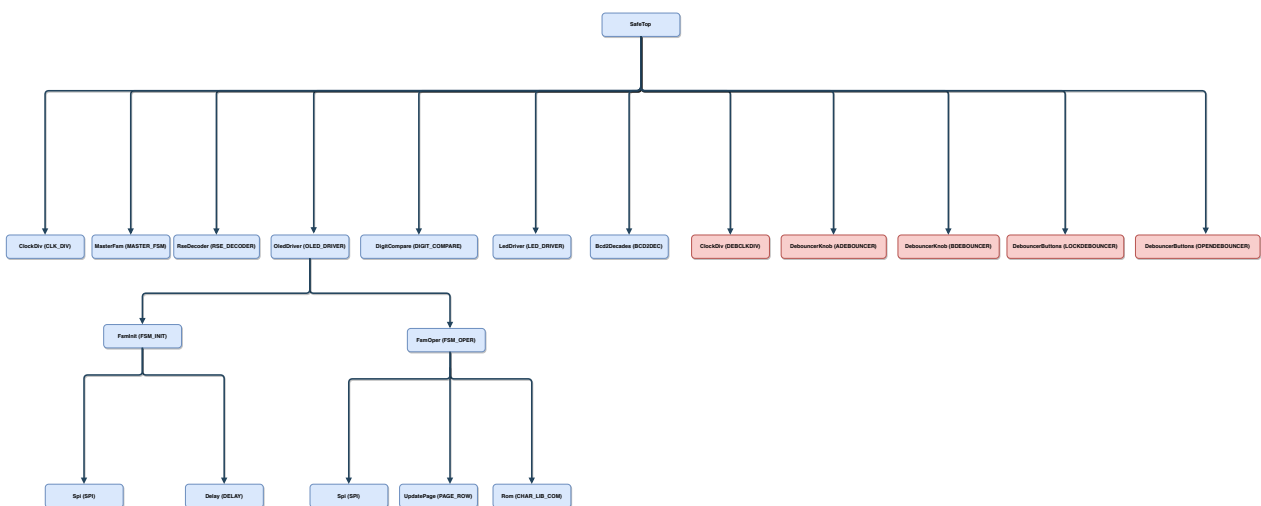
- możliwość zamknięcia sejfu - za pomocą przycisku *Close*
- możliwość ustalenia aktualnej pozycji rygla sejfu - za pomocą przełącznika - zastępuje to czujnik zamknięcia sejfu
- możliwość łatwego zaprogramowania nowego szyfru - zmiana trzech wartości w kodzie

3 Dokumentacja użytkownika

4 Dokumentacja techniczna

4.1 Warstwa Hardware

4.2 Warstwa Software - architektura



Poniższy diagram przedstawia wysokopoziomą architekturę projektu:

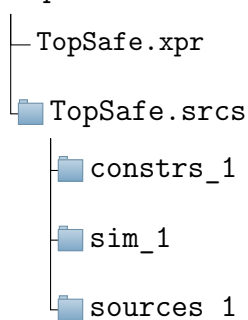
TUTAJ DAC DIAGRAM

Poniższy diagram przedstawia strukturę modułów i połączeń między nimi zastosowaną w projekcie:

TUTAJ DAC DIAGRAM

Wysokopoziomowa struktura katalogów i plików przedstawiona została poniżej:

TopSafe



Struktura wewnętrzna katalogu **constrs_1** jest następująca:

constrs_1

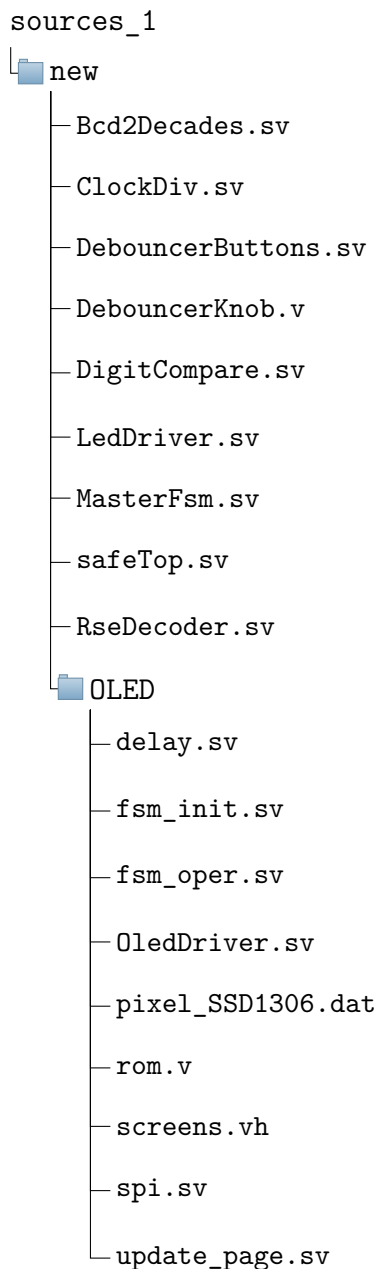
└─ new
 └─ constraints.xdc

Struktura katalogu zawierającego moduły testowe:

sim_1

└─ new
 └─ OLED
 └─ TbOledDriver.sv
 ─ TbBcd2Decades.sv
 ─ TbClkDiv.sv
 ─ TbDebouncerButtons.sv
 ─ TbDelay.sv
 ─ TbDigitCompare.sv
 ─ TbLedDriver.sv
 ─ TbMasterFsm.sv
 ─ TbRseDecoder.v
 ─ TbSafeTop.sv

Struktura katalogu zawierającego kod źródłowy projektu:



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.

- **areDebounceUsed** - 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 Lista modułów 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 sejf
- **RseDecoder** - układ odpowiedzialny za komunikację z pokrętem
- Moduły odpowiedzialne za **obsługę wyświetlacza OLED**:
 - **OledDriver** - główny moduł odpowiedzialny za obsługę wyświetlacza
 - **Delay** - moduł odpowiedzialny za generowanie opóźnień
 - **FsmInit** - moduł odpowiedzialny za przeprowadzenie procedury inicjalizacji wyświetlacza OLED
 - **FsmOper** - moduł odpowiedzialny za wysyłanie danych do wyświetlacza OLED
 - **Rom** - transkoder działający na zasadzie pamięci stałej - odpowiada za dostarczenie *czcionki*
 - **Spi** - implementacja uproszczonego (jednokierunkowego) protokołu SPI
 - **UpdatePage** - moduł powiązany z FsmOper

Do implementacji modułów obsługujących wyświetlacz OLED wykorzystany został w dużej części kod przygotowany w ramach zajęć laboratoryjnych z przedmiotu Języki Opisu Sprzętu. Dalsza część dokumentacji zawiera opis działania najważniejszych modułów projektu.

4.3.3 Moduł SafeTop

4.3.4 Moduł RseDecoder

4.3.5 Moduł Bcd2Decades

4.3.6 Moduł MasterFsm

5 Analiza raportu syntezy

Rozdział zawiera analizę raportu syntezy wygenerowanego przez narzędzie **Xilinx Vivado 2019.1**. Analizie poddane zostały:

- Sposób kodowania stanów
- Użyte zasoby, np.: ROM, komórki logiczne

5.1 Kodowanie stanów

- W większości przypadków zastosowane zostało kodowanie typu **one-hot**, co jest zachowaniem domyślnym dla narzędzi Xilinx
- Dotyczy to następujących modułów: MasterFsm, RseDecoder, Spi, FsmOper, FsmInit
- Tabela przedstawiająca przykładowe kodowanie **one-hot**:
- WSTAWIC TABELKE :D @ARKADIUSZ KASPRZAK
- W przypadku pozostałych maszyn stanów zostało zastosowane kodowanie sekwencyjne. (Delay, UpdatePage, OledDriver)
- Tabela przedstawiająca przykładowe kodowanie **sekwencyjne**:
- WSTAWIC TABELKE :D @ARKADIUSZ KASPRZAK

5.2 Użyte zasoby

- Użyty został jeden blok ROM o rozmiarach 1024x8. Przeznaczony on jest na przechowywanie informacji o czcionce - informacje wczytane z pliku **pixel_SSD1306.dat**
- Cały projekt składa się z **635** komórek logicznych, w tym największą ilość stanowią komórki typu **FDCE** oraz **LUT** z różną ilością argumentów.
- Poniższa tabela przedstawia zużycie komórek dla poszczególnych modułów:
- WSTAWIC TABELKE :D @ARKADIUSZ KASPRZAK

6 Testy

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

6.1 Moduły testujące

Projekt zawiera **10** modułów testowych (tzw. moduły *Testbench*). Umożliwiają one przeprowadzenie symulacji działania modułów projektu. Testy przeprowadzone zostały za pomocą dwóch typów symulacji:

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

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 wartosci sygnalu in1
end

// ...

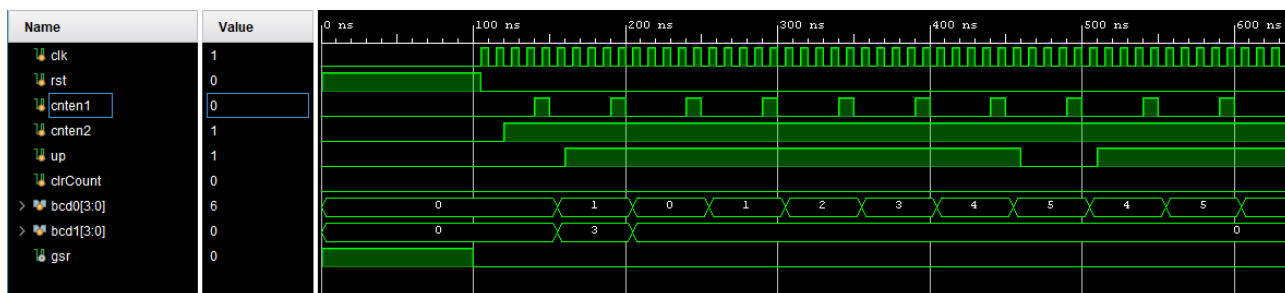
endmodule

```

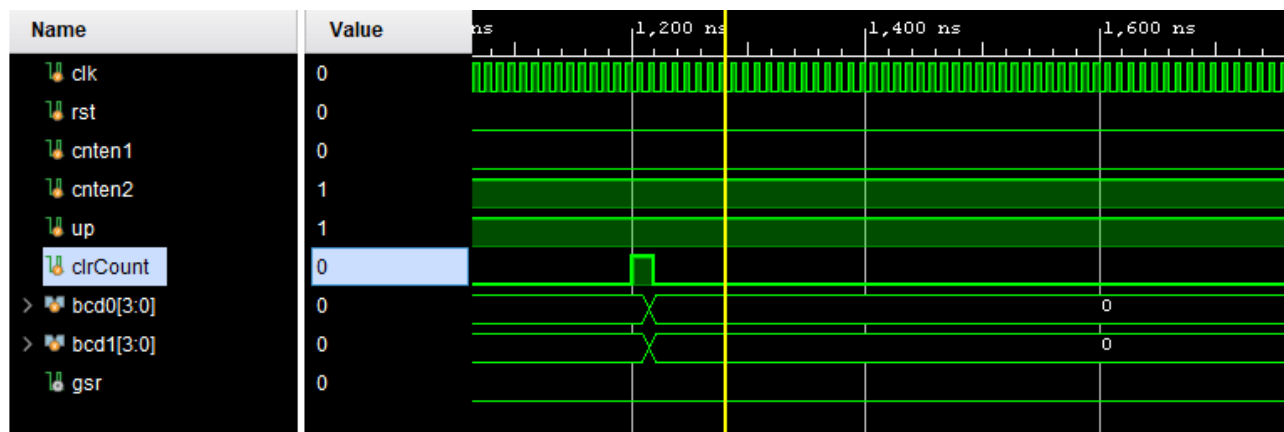
W dalszej części tego podrozdziału omówione zostaną poszczególne moduły testujące oraz wyniki przeprowadzonych symulacji behawioralnych.

6.1.1 Testy modułu SafeTop

6.1.2 Testy modułu Bcd2Decades



Rysunek 1: Tutaj dac opis



Rysunek 2: Tutaj dac opis

- 6.1.3 Testy modułu RseDecoder**
- 6.1.4 Testy modułu MasterFsm**
- 6.1.5 Testy modułu ClkDiv**
- 6.1.6 Testy modułu DebouncerButtons**
- 6.1.7 Testy modułu DigitCompare**
- 6.1.8 Testy modułu OledDriver**
- 6.1.9 Testy modułu Delay**
- 6.2 Testy manualne**
- 7 Możliwe udoskonalenia**