Data: 12.05.2021

**Politechnika Wrocławska**

Układy Cyfrowe i Systemy Wbudowane 2

Dokumentacja projektu:

Organy generujące fale piłokształtne z

wyświetlaniem informacji na kanale tekstowym

Skład grupy:

Artur Sobolewski 248913

Przemysław Rychter 248820

Prowadzący:

Dr inż. Jarosław Sugier

1. **Wprowadzenie**
   1. **Cel projektu**

Celem projektu było wykonanie układu realizującego organki generujące dźwięki w postaci fali piłokształtnych z zakresu jednej oktawy. Obsługa urządzenie miała odbywać się za pomocą klawiatury wysyłającej sygnały przez interfejs PS2. Miało to pozwolić generować odpowiednie tony w zależności od naciśniętego klawisza. Za odtwarzanie dźwięków odpowiadałby podłączony do płyty głośniczek. Dodatkową formą komunikacji z użytkownikiem ma być wyświetlanie informacji dotyczącej wysokości i długości dźwięku na kanale tekstowym VGA.

* 1. **Wykorzystany sprzęt i narzędzia**

Docelowym sprzętem do zaprogramowania była płyta FPGA Spartan-3E Starter Kit. Wykorzystywanym środowiskiem było oprogramowanie Xilinx ISE, które pozwala na realizację kodu w języku opisu sprzętu VHDL oraz jego kompilację dostosowaną do podanej wcześniej płyty i końcowo wykonywać symulacje układu. Do projektu zostały przewidziane dodatkowe urządzenia: klawiatura wyposażona w interfejs PS2, głośniczek oraz wyświetlacz odbierający informacje z portu VGA.

1. **Realizacja projektu**
   1. **Top module schemat**

**TODO**

* 1. **Moduły składowe**
     1. **dac\_signal\_generator**

Moduł dac\_signal\_generator odpowiada za generowanie fali piłokształtnej i sygnałów sterujących, w formacie dostosowanym do danych jakie przyjmuje modułu DACWrite (fala - wartości liczb całkowitych w postaci wektora 12-bitoweg). Wartości przyjmowane na wektorze Key określają częstotliwość fali, jaka ma być generowana. Praca tego modułu dostosowana jest do częstotliwości z jaką pracuje wykorzystana płyta Spartan – 50 MHz – w oparciu o nią dobierane są opóźnienia w zmianie wartości wychodzącej, dzięki czemu otrzymujemy określone częstotliwości fal.

**Schemat dac_signal_generator

Opis wygenerowany automatycznie**

Wejścia:

* Rdy – pełni rolę sygnału sterującego, które powoduje pracę lub spoczynek modułu,
* Clk – sygnał zegarowy,
* Key(3:0) – wartości określające częstotliwość fali do wygenerowania,

Wyjścia:

* Start – impuls wysyłany do modułu DACWrite – informujący by ten przyjął przesyłane dane,
* Command(3:0) – wektor określający pracę modułu DACWrite,
* Address(3:0) – wektor określający, na który port wyjściowy ma być wysyłany sygnał analogowy z modułu DACWrite,
* Data(11:0) – wektor, którego określa wartość sygnału w danym czasie.

Opis realizacji modułu:

Na samym początku architektura modułu dac\_signal\_generator definiuje zmienne:

* clks\_delay – liczba określające ile taktów zegara należy odczekać pomiędzy kolejnymi zmianami wartości sygnału (bazowa wartość 999 – pozwala uzyskać częstotliwość równą 1 kHz). Konkretne częstotliwości są możliwe do uzyskania dzięki kontrolowaniu częstotliwości inkrementacji wartości przechowywanej w zmiennej var – przed każdą inkrementacją proces odczekuje zadaną liczbę taktów, których ilość jest zależna od wartości na wejściu Key. Zależność ta została stabelaryzowana w instrukcji with-select.
* clk\_counter – zmienna, która realizuje zliczanie taktów
* var – zmienna inkrementowana, której wartość przekazywana jest na wyjście Data.

W procesie został zaimplementowany licznik modulo działający w maksymalnym zakresie liczb całkowitych 12-bitowych. Wartość zwiększana jest o dziesiętną wartość 82, po zliczeniu wymaganej liczby okresów. Zależność liczby taktów zegara do odczekania od częstotliwości fali, jaka ma zostać wygenerowana wyznaczana była następująco:

Jeśli długość taktu sygnału zegarowego płyty: , liczba kolejnych wartości składających się na jeden okres fali piłokształtnej: , to długość okresu fali piłokształtnej:

,

Gdzie:

T – długość okresu,

x – liczba taktów zegara, jaką należy odczekać (we wzorze występuje + 1, ponieważ jest to dodatkowy takt potrzebny na inkrementację zmiennej)

Następnie jeśli: to po przekształceniach otrzymujemy, zależność od częstotliwości:

**entity** dac\_signal\_generator **is**

**Port** **(** Rdy **:** **in** STD\_LOGIC**;**

Clk **:** **in** STD\_LOGIC**;**

Key **:** **in** STD\_LOGIC\_VECTOR **(**3 **downto** 0**);**

Command **:** **out** STD\_LOGIC\_VECTOR **(**3 **downto** 0**);**

Address **:** **out** STD\_LOGIC\_VECTOR **(**3 **downto** 0**);**

Data **:** **out** STD\_LOGIC\_VECTOR **(**11 **downto** 0**);**

Start **:** **out** STD\_LOGIC**);**

**end** dac\_signal\_generator**;**

**architecture** Behavioral **of** dac\_signal\_generator **is**

**signal** clks\_delay **:** integer **:=** 999**;**

**signal** clk\_counter **:** integer **:=** 0**;**

**signal** var **:** unsigned **(**11 **downto** 0**)** **:=** X"000"**;**

**begin**

**with** Key **select** clks\_delay **<=**

955 **when** "0000"**,** -- 1046.50 c [ HZ ]

901 **when** "0001"**,** -- 1108.73 cis

850 **when** "0010"**,** -- 1174.66 d

802 **when** "0011"**,** -- 1244.51 dis

758 **when** "0100"**,** -- 1318.51 e

715 **when** "0101"**,** -- 1396.91 f

675 **when** "0110"**,** -- 1479.98 fis

637 **when** "0111"**,** -- 1567.98 g

601 **when** "1000"**,** -- 1661.22 gis

567 **when** "1001"**,** -- 1760.00 a

535 **when** "1010"**,** -- 1864.66 b

505 **when** "1011"**,** -- 1975.53 h

0 **when** **others;**

Command **<=** "0011"**;**

Address **<=** "0000"**;**

**process(**Clk**)**

**begin**

**if** **rising\_edge(**Clk**)** **and** Rdy **=** '1' **then**

**if** clks\_delay **/=** 0 **then** --

**if** clk\_counter **<** clks\_delay **then**

clk\_counter **<=** clk\_counter **+** 1**;**

Start **<=** '0'**;**

**else**

Start **<=** '1'**;**

**if** var **<** X"FAD" **then**

var **<=** var **+** 82**;**

**else**

var **<=** X"000"**;**

**end** **if;**

clk\_counter **<=** 0**;**

**end** **if;**

**else**

var **<=** X"000"**;**

clk\_counter **<=** 0**;**

**end** **if;**

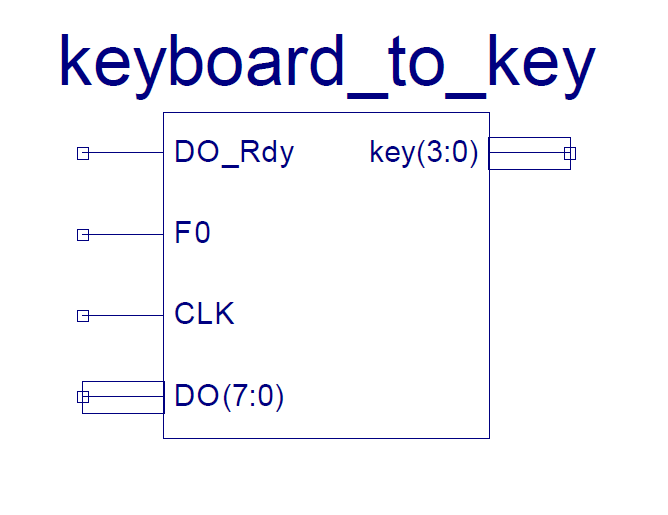
**end** **if;**

**end** **process;**

Data **<=** STD\_LOGIC\_VECTOR**(**var**);**

**end** Behavioral**;**

* + 1. **keyboard\_to\_key**

****

* 1. **TODO**