Układy cyfrowe i systemy wbudowane 2
Projekt - Test czasu reakcji

Autorzy:
Filip Przygoński (248892)
Radosław Łodziński (249455)

Politechnika Wrocławska
Wydział Elektroniki
Prowadzący: *dr inż. Jacek Mazurkiewicz*

Spis treści

1 Założenia wstępne programu

Po uruchomieniu wyświetla się ekran startowy, witający gracza i czekający na wciśnięcie dowolnego przycisku na klawiaturze. Po rozpoczęciu na ekranie pojawią się czerwone światła, zapalone na losową długość czasu. Po ich zgaśnięciu, uruchamia się stoper i program czeka na reakcję użytkownika. Gdy użytkownik wciśnie dowolny przycisk, stoper się zatrzymuje, a na ekranie wyświetla się czas reakcji. Po wciśnięciu dowolnego przycisku, można zacząć grę od nowa.

2 Aktualny stan projektu

Projekt posiada 3 skończone własnoręcznie wykonane przez nas moduły - do mierzenia czasu reakcji, do odliczania czasu do zgaśnięcia świateł, oraz do generowania pseudolosowych liczb. Są one wykorzystywane w głównym module gry, który łączy wszystko w całość. W planach mieliśmy wykorzystanie modułów dr Sugiera do odbioru sygnałów z klawiatury PS/2 oraz do wyświetlania świateł startowych i tekstu na ekranie, jednakże mieliśmy z nimi problemy i spróbowaliśmy je zastąpić innymi tymczasowymi rozwiązaniami.

3 Moduły

3.1 Timer

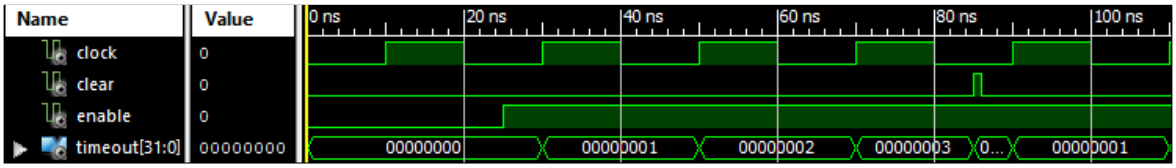
Moduł Timer działa jak stoper, jest on wykorzystywany do zmierzenia czasu reakcji użytkownika. Czas jest mierzony za pomocą zliczania narastających zboczy zegara.

Moduł posiada porty:

- *Clock* - wejście zegarowe.
- *Clear* - wejście asynchronicznie zerujące czas, gdy jest ustawione na ’1’.
- *Enable* - wejście włączające moduł, stoper działa, gdy jest ustawione na ’1’.
- *TimeOut* - wyjście - zmierzony czas (zliczone narastające zbocza zegara) jako wektor 32-bitowy.

Listing 1: Kod modułu Timer

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity Timer is
    Port (  Clock : in  STD_LOGIC;
            Clear : in STD_LOGIC;
            Enable : in STD_LOGIC;
            TimeOut : out  STD_LOGIC_VECTOR (31 downto 0));
end Timer;

architecture Behavioral of Timer is
   signal TimeInternal : UNSIGNED(31 downto 0) := (others => '0');
begin

    process(Clock, Clear)
    begin
        if Clear = '1' then
            TimeInternal <= (others => '0');
        elsif rising_edge(Clock) then
            if Enable = '1' then
                TimeInternal <= TimeInternal + 1;
            end if;
        end if;
    end process;

    TimeOut <= STD_LOGIC_VECTOR(TimeInternal);

end Behavioral;
```



Rysunek 1: Symulacja modułu Timer

Jak widać na symulacji, na każdym narastającym zboczu zegara *TimeOut* zwiększa się o jeden, ale tylko wtedy gdy *Enable* jest ustawione na '1'. *Clear* działa asynchronicznie i przywraca *TimeOut* do zera.

3.2 Countdown

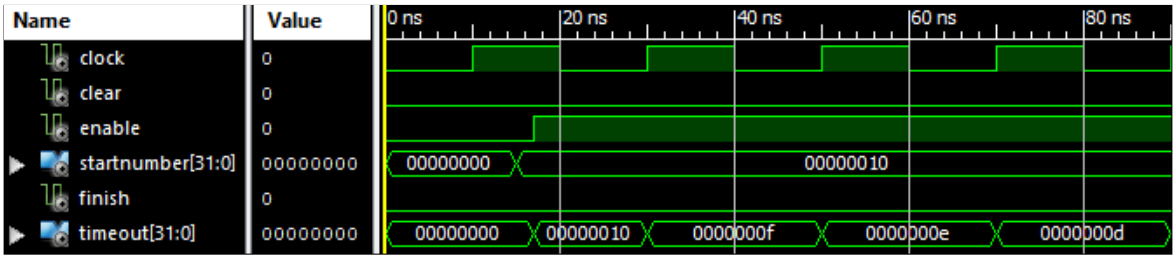
Moduł odliczający do zgaśnięcia "świateł" na ekranie. Po włączeniu, załadowuje liczbę podaną na wejściu i zlicza od tej liczby do zera na narastających zboczach zegara. Gdy licznik osiągnie zero, sygnalizuje to na wyjściu.

Moduł posiada porty:

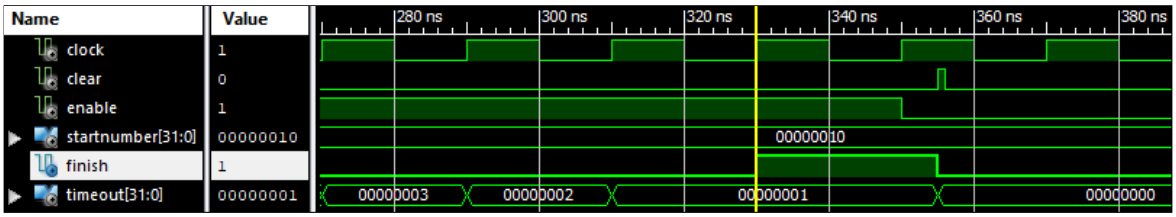
- Clock - wejście zegarowe.
- Clear - wejście asynchronicznie zerujące czas, gdy jest ustawione na '1'.
- Enable - wejście włączające moduł, licznik działa, gdy jest ustawione na '1'. Gdy to wejście zostanie przełączone na '1' (narastające zbocze), do modułu ładowana jest liczba z wejścia StartNumber.
- StartNumber - wejście - liczba, od której ma się zacząć zliczanie (wektor 32-bitowy).
- TimeOut - wyjście - aktualny czas (wektor 32-bitowy).
- Finish - wyjście sygnalizujące zakończenie odliczania.

Listing 2: Kod modułu Countdown

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;


entity Countdown is
    Port (  Clock : in  STD_LOGIC;
            Clear : in  STD_LOGIC;
            Enable : in  STD_LOGIC;
            StartNumber : in  STD_LOGIC_VECTOR (31 downto 0);
            TimeOut: out STD_LOGIC_VECTOR (31 downto 0);
            Finish : out  STD_LOGIC);
end Countdown;

architecture Behavioral of Countdown is
    signal TimeInternal : UNSIGNED(31 downto 0) := x"00000000";
    signal FinishInternal : STD_LOGIC := '0';
begin

    process(Clock, Clear, Enable)
    begin
        if Clear = '1' then
            TimeInternal <= x"00000000";
            FinishInternal <= '0';
        -- Enable wczytuje liczbe ze 'StartNumber'
        -- i uruchamia odliczanie
        elsif rising_edge(Enable) then
            TimeInternal <= UNSIGNED(StartNumber);
            FinishInternal <= '0';
        elsif rising_edge(Clock) then
            if Enable = '1' then
                -- nie zero poniewaz bedzie
                -- opoznienie jednego ticku zegara
                if TimeInternal = 1 then
                    FinishInternal <= '1';
                else
                    TimeInternal <= TimeInternal - 1;
                    FinishInternal <= '0';
                end if;
            end if;
        end if;
    end process;

    Finish <= FinishInternal;
    TimeOut <= STD_LOGIC_VECTOR(TimeInternal);

end Behavioral;
```



Rysunek 2: Symulacja modułu Countdown (załadowanie liczby od której będzie zliczane)



Rysunek 3: Symulacja modułu Countdown (koniec zliczania, sygnalizacja na wyjściu *Finish*)

Jak widać na symulacjach, moduł poprawnie załadowuje nową liczbę, oraz sygnalizuje koniec zliczania na wyjściu *Finish*.

3.3 RNG

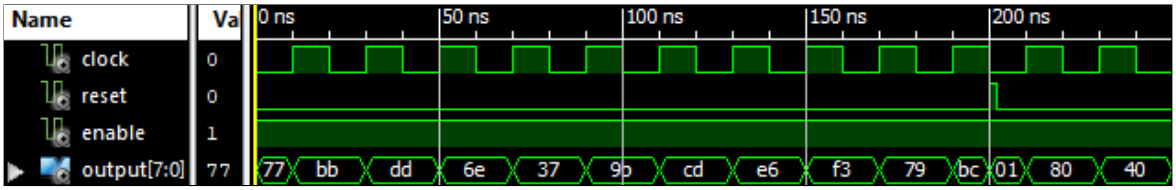
Moduł RNG (Random Number Generator) jest wykorzystywany do generowania pseudolosowych liczb 8-bitowych. Został on stworzony za pomocą rejestru przeswuającego z liniowym sprzężeniem zwrotnym (*LFSR*).

Moduł posiada porty:

- *Clock* - wejście zegarowe.
- *Reset* - wejście - asynchroniczny reset.
- *Enable* - wejście włączające moduł, generowanie losowych liczb działa, gdy jest ustawione na '1'.
- *Output* - wyjście - wygenerowana liczba 8-bitowa.

Listing 3: Kod modułu RNG

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity RNG is
    Port (  Clock : in  STD_LOGIC;
            Reset : in  STD_LOGIC;
            Enable : in  STD_LOGIC;
            Output : out  STD_LOGIC_VECTOR (7 downto 0));
end RNG;

architecture Behavioral of RNG is
    signal OutputTemp: STD_LOGIC_VECTOR(7 downto 0) := x"77";
begin

process(clock)
    variable Temp: STD_LOGIC := '0';
    begin
        if (Reset = '1') then
                OutputTemp <= x"01";
        elsif rising_edge(Clock) then
            if Enable = '1' then
                Temp := OutputTemp(4) xor OutputTemp(3) xor
                OutputTemp(2) xor OutputTemp(0);

                OutputTemp <= Temp & OutputTemp(7 downto 1);
            end if;
        end if;
end process;

Output <= OutputTemp;

end Behavioral;
```



Rysunek 4: Symulacja modułu RNG

Jak widać na symulacji modułu, na każdym narastającym zboczu zegara na wyjście jest podawana kolejna wygenerowana pseudolosowa liczba. Reset zawsze przywraca moduł do liczby 1.

3.4 PS2_Kbd - moduł dr Sugiera

Moduł ten byłby wykorzystywany do odbierania kodów wysyłanych przez klawiaturę PS/2 od użytkownika. Z powodu trudności z gotowymi modułami oraz brakiem sprzętu został on tymczasowo zastąpiony przez prosty sygnał wewnętrzny, który sygnalizuje, czy na klawiaturze zadziała się jakaś „akcja".

Link do modułu

3.5 VGAtxt48x20 - moduł dr Sugiera

Moduł ten byłby wykorzystywany do wyświetlania tekstu na ekranie startowym, do wyświetlania „świateł startowych", oraz do wyświetlania uzyskanego czasu reakcji na ekranie. Z powodu trudności z gotowymi modułami oraz brakiem sprzętu, został on tymczasowo zastąpiony przez komentarze w kodzie mówiące, jaka informacja powinna się pojawić w danym momencie na ekranie.

Link do modułu

4 Główny moduł - test czasu reakcji

4.1 Instancje komponentów

Główny moduł gry wykorzystuje wszystkie zadeklarowane przez nas wcześniej komponenty. Na przykładzie generatora liczb pseudolosowych pokażemy, jak są one instancjonowane w kodzie.

Listing 4: Komponent (umieszczony przed rozpoczęciem architektury)

```
COMPONENT RNG
PORT(
     Clock : IN std_logic;
     Reset : IN std_logic;
     Enable : IN std_logic;
     Output : OUT std_logic_vector(7 downto 0)
    );
END COMPONENT;
```

Listing 5: Instancja komponentu oraz mapowanie portów (w architekturze modułu)

```
-- Instantiate RNG
_rng: RNG PORT MAP (
    Clock => Clock,
    Reset => ResetRNG,
    Enable => EnableRNG,
    Output => OutputRNG
);
```

W mapowaniu portów, porty po lewej to porty komponentu, natomiast po prawej porty to sygnały wewnętrzne modułu gry, o których więcej w następnej sekcji.

4.2 Sygnały wewnętrzne

Moduł gry posiada wiele sygnałów wewnętrznych.

Listing 6: Sygnały dla modułu Countdown

```
-- Inputs
signal ClearCountdown : std_logic := '0';
signal EnableCountdown : std_logic := '0';
signal StartNumberCountdown : std_logic_vector(31 downto 0) :=
    (others => '0');

-- Outputs
signal FinishCountdown : std_logic;
signal TimeOutCountdown : std_logic_vector(31 downto 0);
```

Listing 7: Sygnały dla modułu RNG

```
--Inputs
signal ResetRNG : std_logic := '0';
signal EnableRNG : std_logic := '1'; -- generowanie losowych liczb
-- bedzie zawsze wlaczone

--Outputs
signal OutputRNG : std_logic_vector(7 downto 0);
```

Listing 8: Sygnały dla modułu Timer

```
--Inputs
signal ClearTimer : std_logic := '0';
signal EnableTimer : std_logic := '0';

--Outputs
signal TimeOutTimer : std_logic_vector(31 downto 0);
```

Listing 9: Pozostałe sygnały nienależące do konkretnych modułów

```
signal Clock : std_logic := '0';
signal KeyboardInput : std_logic := '0';
signal GameState : StateType;
```

- Clock - sygnał zegarowy, wykorzystywany we wszystkich modułach, oraz oczywiście w samym module gry.
- KeyboardInput - sygnał wskazujący czy z klawiatury była jakaś 'akcja', równoznaczny z DO_Rdy z modułu PS2_Kbd
- GameState - sygnał wskazujący na obecny stan gry, więcej w następnej sekcji.

4.3 Stan gry

Gra jest maszyną stanów.

Listing 10: Stany gry (typ Enum)

```
type StateType is (StartScreen, BeginCountdown, LightsOn, TimerOn,
    LightsOut, EndScreen);
signal GameState : StateType;
```

1. StartScreen - wyświetlenie ekranu startowego i czekanie na rozpoczęcie gry przez użytkownika
2. BeginCountdown - wylosowanie liczby i rozpoczęcie odliczania
3. LightsOn - wyświetlenie na ekranie świateł, czekanie na koniec odliczania
4. TimerOn - rozpoczęcie liczenia czasu
5. LightsOut - zgaśniecie świateł, czekanie na reakcję użytkownika
6. EndScreen - użytkownik zareagował, kończymy liczenie czasu, wyświetlamy ekran końcowy z czasem

4.4 Działanie modułu

Listing 11: Główny proces

```
process(Clock)
begin
    if rising_edge(Clock) then
        if ResetGame = '1' then
            GameState <= StartScreen;
        else
            case GameState is
                when StartScreen =>
                    -- [...]

                when BeginCountdown =>
                    -- [...]

                when LightsOn =>
                    -- [...]

                when TimerOn =>
                    -- [...]

                when LightsOut =>
                    -- [...]

                when EndScreen =>
                    -- [...]
            end case;
        end if;
    end if;
end process;
```

Ponieważ cały proces jest bardzo długi, fragmenty kodu dla każdego stanu pokażemy w osobnych listingach.

4.4.1 StartScreen

```
when StartScreen =>
    -- wyswietlenie tekstu

    -- jesli uzytkownik cos kliknal to przechodzimy do gry
    if KeyboardInput = '1' then
        GameState <= BeginCountdown;
        -- wyzerowanie aktywnosci klawiatury
        KeyboardInput <= '0';
    end if;
```

4.4.2 BeginCountdown

```
when BeginCountdown =>
    -- wstawienie losowej liczby do modulu Countdown
    StartNumberCountdown <= OutputRNG & OutputRNG &
        OutputRNG & OutputRNG;
    -- wlaczenie modulu Countdown
    ClearCountdown <= '0';
    EnableCountdown <= '1';
    GameState <= LightsOn;
```

4.4.3 LightsOn

```
when LightsOn =>
    -- wyswietlenie swiatel

    -- czekanie na koniec countdown
    if FinishCountdown = '1' then
        -- jezeli koniec to wylaczamy countdown
        EnableCountdown <= '0';
        ClearCountdown <= '1';
        -- wyczyszczenie timera przed wlaczeniem go
        ClearTimer <= '1';
        GameState <= TimerOn;
    end if;
```

4.4.4 TimerOn

```
when TimerOn =>
    ClearTimer <= '0';
    EnableTimer <= '1';
    GameState <= LightsOut;
```

4.4.5 LightsOut

```
when LightsOut =>
    -- czekanie na reakcje uzytkownika
    if KeyboardInput <= '1' then
        EnableTimer <= '0';
        GameState <= EndScreen;
    end if;
```

4.4.6 EndScreen

```
when EndScreen =>
    -- wyswietlenie ekranu koncowego z czasem (TimeOutTimer)

    -- klikniecie = zaczecie gry od nowa
    if KeyboardInput <= '1' then
        GameState <= StartScreen;
    end if;
```

5 Opis działania

W obecnym stanie, projekt nie jest używalny przez zwykłego użytkownika. Mamy jednakże szkielet, który pokazuje zarys działania. Użytkownik klikając dowolne przyciski na klawiaturze może, w zależności od stanu gry: rozpocząć grę, zareagować na sygnał startu, oraz spróbować jeszcze raz.

6 Podsumowanie

Główny cel projektu, jakim było zmierzenie czasu reakcji został osiągnięty. Po drodze napotkaliśmy parę problemów. Najtrudniejszym z nich były trudności związane z korzystaniem z modułów dr Sugiera. Moduł klawiaturowy udało nam się zastąpić sygnałem wewnętrznym KeyboardInput, natomiast nie znaleźliśmy sposobu na wyświetlanie świateł z uwagi na brak dostępnego sprzętu. Z tego powodu w głównym procesie wyświetlanie tekstu było ukazane tylko w formie komentarza. Właśnie ze względu na brak dostępu do układu, klawiatury PS2, ani monitora z wejściem VGA stworzenie takiej gry jest znacznie utrudnione, a testowanie praktycznie niemożliwe.

7 Źródła

1. *Rejestr przesuwający z liniowym sprzężeniem zwrotnym*
2. *Moduły dr Sugiera*