KAUE
Projekt cyfrowy
Sprawozdanie

Paweł Lipiór

24 kwietnia 2020

1 Wstęp

Zadaniem projektowym było zaprojektowanie modułu `counter` realizującego licznik 4-bitowy w kodzie Gray'a z 2-bitowym wejściem sterującym `ctrl`. Stan wejścia sterującego określał zachowanie licznika:

- `ctrl` = 0 - licznik liczy w tył
- `ctrl` $\neq$ 0 - `ctrl` traktujemy jak liczbę zapisaną w kodzie U2 i dodajemy ją do aktualnej wartości licznika

Do zrealizowanych zadań należało także napisanie modułu `test_bench` służącego do testowania napisanego modułu `counter`.
Dodatkowym celem, który starano się zrealizować w projekcie, było napisanie kodu tak, aby względnie prosto możliwe było rozszerzenie bitowe licznika.
Realizacje testów modułu zostały zapisane w plikach `.vcd`. Do ich wyświetlenia posłużono się programem `GTKWave`. Rysunki obrazujące przebiegi zapisano następnie jako pliki graficzne i wstawiono do sprawozdania.

2 Realizacja licznika

2.1 Konwersja kodu binarnego na kod Gray'a

W trakcie projektowania licznika stwierdzono iż zrealizowanie modułu w kodzie binarnym jest znacznie prostsze, gdyż komputery posługują się arytmetyką binarną. Z tego powodu postanowiono realizować arytmetykę licznika w kodzie binarnym, a dopiero na wyjściu licznika zamienić wynik w kodzie dwójkowym na kod Gray'a.

W celu opracowania metody konwersji N-bitowej liczby w kodzie binarnym na kod Gray'a posłużono się tabelą zawierającą liczby w kodzie 3-bitowym binarnym i kodzie Gray'a.

Tabela 1: Tablica przedstawiająca zapis liczb naturalnych w kodzie binarnym i Gray'a

N	Zapis binarny $[a_2][a_1][a_0]$	Zapis Gray'a $[b_2][b_1][b_0]$
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Możemy zauważyć, że konwersja z 3-bitowego kodu binarnego na 3-bitowy kod Gray'a polega na:

- przepisaniu bitu a_2 na b_2
- jeśli bit a_2 oraz a_1 są różne to zapisujemy bit b_1 jako 1, w innym wypadku 0
- jeśli bit a_1 oraz a_0 są różne to zapisujemy bit b_0 jako 1, w innym wypadku 0

Prowadzi to do następującego przepisu konwersji z 3-bitowego kodu binarnego na 3-bitowy kod Gray'a:

$$[a_2]\,[a_1]\,[a_0] \rightarrow [a_2]\,[XOR(a_2, a_1)]\,[XOR(a_1, a_0)]$$

Powyższy przepis możemy uogólnić na N-bitowy kod:

$$[a_{N-1}]\,[a_{N-1}]\,\ldots\,[a_0] \rightarrow [a_{N-1}]\,[XOR(a_{N-2}, a_{N-2})]\,\ldots\,[XOR(a_1, a_0)]$$

Dzięki temu przepisowi możemy zrealizować licznik dla dowolnej liczby N bitów.

2.2 Realizacja modułu licznika

Moduł `counter` zrealizowano w języku Verilog. Kod programu zamieszczono poniżej.

Moduł licznika

```
module counter(clk, reset, ctrl, out);

        parameter N = 4;        //Parametr. Domyślnie 4 bit out, N >= 2
        input clk, reset;        //Zegar oraz reset
        input [1:0] ctrl;        //Sterowanie 2-bitowe
        output [N-1:0] out;        //Wyjście N-bitowe, kod Gray'a
        reg [N-1:0] tmp;        //Zmienna pomocnicza, kod binarny

        always @(posedge clk, negedge reset) begin
                //Sprawdzenie resetu
                if (reset == 0) begin
                        tmp <= 0;
                end else begin
                                //Sterowanie na licznik do tyłu
                                if (ctrl == 0) begin
```

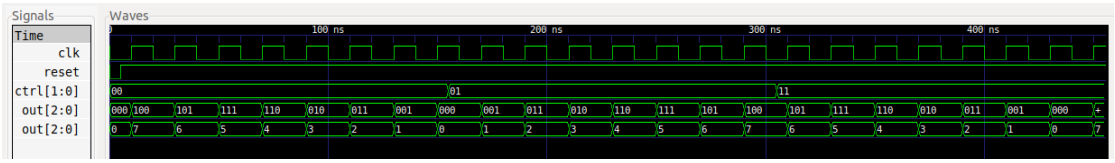
```
                                        tmp <= tmp - 1;
                                    //Sterowanie na licznik sterowany ctrl w kodzie U2
                                    end else begin
                    //Użycie modyfikatora £signed umożliwia działania na liczbach w kodzie U2
                                        tmp <= $signed(tmp) + $signed(ctrl);
                                    end
                    end
            end
            //Konwersja z N-bitowego kodu binarnego na N-bitowy kod Gray'a
            assign out = {tmp[N-1], tmp[N-1:1] ^ tmp[N-2:0]}; // ^ - operator XOR
            endmodule
```

Moduł testowy

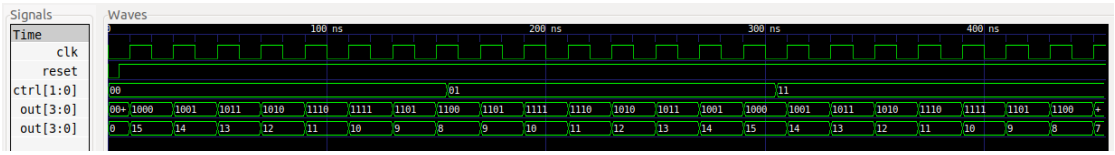
```
`timescale 1ns/100ps

module test_bench;

        parameter N = 4;        //Parametr. Domyślnie 4 bit out, N >= 2
        //Input
        reg clk = 0;                //Zegar
        reg reset;                      //Reset
        reg [1:0] ctrl = 0;         //Sterowanie 2-bitowe

        //Output
        wire [N-1:0] out;         //Wyjście N-bitowe

        always
                # 10 clk = ~clk;        //Zmiana zegara, okres 20 ns

                counter count(clk, reset, ctrl, out);        //Wywołanie modułu counter

        initial begin

                $dumpfile("counter.vcd");
                $dumpvars(0, test_bench);

                reset = 0;
                #5 reset = 1;                //Ustawienie resetu
                #150 ctrl = 2'b01;
                #150 ctrl = 2'b11;
                #150 $finish;
        end

endmodule
```

3 Wyniki symulacji

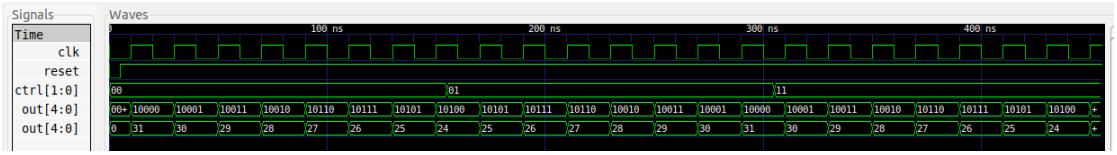
w celu zweryfikowania poprawności zrealizowanego modułu `counter` przy pomocy `test_bech` dokonano symulacji działania licznika dla zmieniającego się po pewnym czasie działania wejścia sterującego `ctrl`. Dokonano podstawowych testów, które przy tak prostej zasadzie działania licznika, powinny po ich przejściu zapewnić o poprawność realizacji modułu licznika. Wyniki przebiegów dla 3, 4 oraz 5-bitowego wyjścia `out` zamieszczono na rysunkach 1, 2 oraz 3. Na przebiegach zaznaczono wyjście `out` zarówno w formie kodu binarnego jak i formie dziesiętnej uwzgledniając filtr z kodu Gray'a.



Rysunek 1: Przebiegi dla licznika 3-bitowego



Rysunek 2: Przebiegi dla licznika 4-bitowego



Rysunek 3: Przebiegi dla licznika 4-bitowego

Po analizie każdego z powyższych rysunków można stwierdzić iż licznik działa prawidłowo dla różnych podanych wejść sterujących N-bitowego licznika.

4 Podsumowanie

W trakcie trwania projektu zrealizowano wszystkie postawione wymagania dotyczące realizowanego licznika oraz modułu testowego. Napisany kod jest bardzo łatwy w modyfikacji dla licznika N-bitowego. Skalowalność sterowania jest równie łatwa do osiągnięcia przez co może być rozszerzone do sterowania N-bitowego.

Po zaprogramowaniu modułu licznika zrealizowano moduł testowy dla licznika. Zaimplementowany kod pozwolił na podstawowe testy poprawności rozwiązania. Pomimo swojej trywialności, ze względu na prostotę modułu licznika, wykonane testy zapewniają, że licznik został zaprogramowany poprawnie.

W trakcie trwania projektu zapoznano się w dużym stopniu ze składnią języka `Verilog`. Wykazano się umiejętnością zaprojektowania i zaprogramowania układu sekwencyjnego. Pomyślnie zaprojektowano i wykonano testy realizowanego modułu `counter`. Kolejno poznano ogólny przepis na zamianę liczby zapisanej w N-bitowym kodzie binarnym na N-bitowy kod Gray'a.