AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI
INSTYTUT ELEKTRONIKI

PROJEKT SYSTEMY DEDYKOWANE W UKŁADACH PROGRAMOWALNYCH

CENZOR

AUTORZY: Michał Czwórnóg, Szymon Adamus
KIERUNEK STUDIÓW: Elektronika i Telekomunikacja
TYP STUDIÓW: Stacjonarne
PROWADZĄCY: dr hab. inż. Paweł Russek

Kraków, 2021

Spis treści

1. Założenia

Celem projektu było stworzenie systemu wbudowanego oraz aplikacji w języku C, które umożliwiałyby cenzorowanie podawanego tekstu. Tekst w aplikacji jest podawany poprzez konsolę, a jako wyjście zostaje wyświetlony tekst, gdzie zamiast słów przeznaczonych do cenzorowania (które można dowolnie zmieniać) zostają wyświetlone znaki "#".

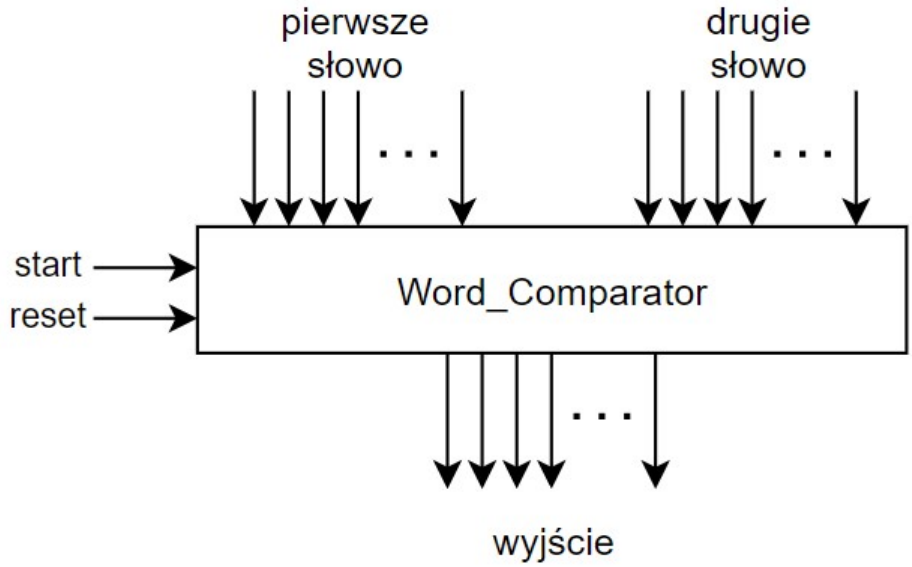
Założenia jakie zostały przyjęte podczas realizacji zadania to:

- Tekst nie powinien zawierać znaku "#", jest on wykorzystany do wykreślania słów,
- Jeżeli wykreślamy słowo „A” i zostanie podane słowo „ABCA” to zostanie ono ocenzurowane w następujący sposób: „#BC#”,
- Możliwość zmiany słów, które są wykreślane podczas działania systemu,
- Maksymalna ilość wykreślanych słów wyniesie 10, natomiast długość tych słów wynosie 15 znaków (takie złożenie jest potrzebne, aby na poziomie hardware‘u przygotować odpowiednią pamięć dla słów przeznaczonych do cenzorowania),
- Wykreślanie słów będzie wrażliwe na wielkość liter.

2. Hardware

2.1. Word Comparator

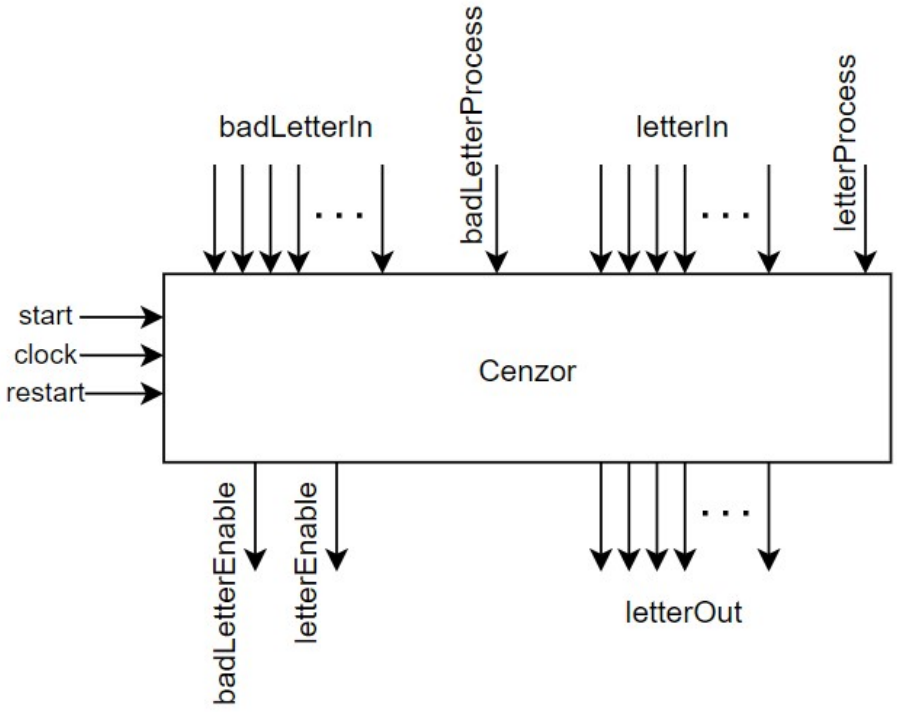
Pierwszym stworzonym elementem przy pomocy języka systemverilog został stworzony element umożliwiający porównywanie dwóch słów wejściowych o zadanej parametrycznie długości. Na wyjście takiego układu zostaje wystawiony rejestr o długości porównywanych słów. W tym rejestrze zostaje ustawiona wartość "1" wtedy i tylko wtedy, gdy litery słów wejściowych (porównanie każdej z liter z osobna) są takie same oraz dany bajt nie jest ustawiony jako "wild card" (czyli może być dowolnym znakiem). Sygnałem wyjściowym będzie sekwencja zer oraz jedynek.



Rysunek 2.1: Schemat bloku Word_Comparator

2.2. Cenzor

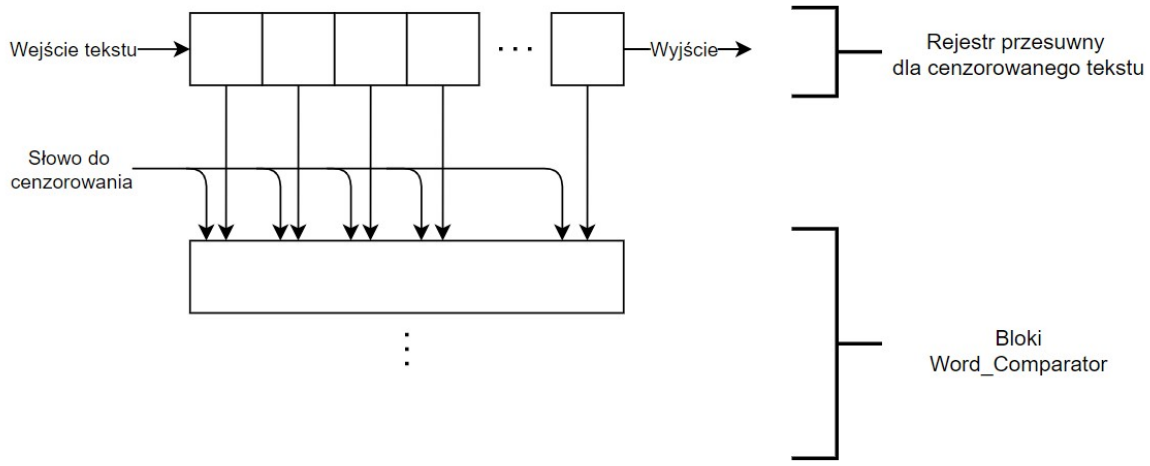
Docelowym elementem, który należało stworzyć, był blok o nazwie Cenzor. Został on pokazany na rysunku 2.2. Blok posiada dwa rejestry przesuwne, z których pierwszy jest odpowiedzialny za pamiętanie słów do cenzorowania, natomiast drugi służy jako rejestr dla cenzorowanego teksu. Tekst/słowa w obu wypadkach podawane są jako bajty odpowiadające literom.



Rysunek 2.2: Schemat bloku Cenzor

Blok stworzony jest w oparciu o wcześniej opisany Word_Comparator. W projekcie założono, że będziemy mieć maksymalnie 10 słów o długości 15 znaków (te zmienne zostały sparametryzowane, więc w innym projekcie można zmienić te wartości). Dla każdego słowa został wygenerowany osobny blok Word_Comparator, który na jedno z wejść przyjmuje jedno ze słów do cenzorowania, drugie wejście jest takie samo dla każdego bloku i jest ono rejestrem przesuwnym cenzorowanego tekstu. Wyjście z bloków Word_Comparator jest połączone i jeżeli, któryś z bitów jest równy "1" dla znaku, zostaje ona zastąpiony znakiem "#" (zostaje ocenzurowany). Dodatkowymi rzeczami w tym elemencie jest kod odpowiedzialny za odpowiednie przesuwanie rejestrów.

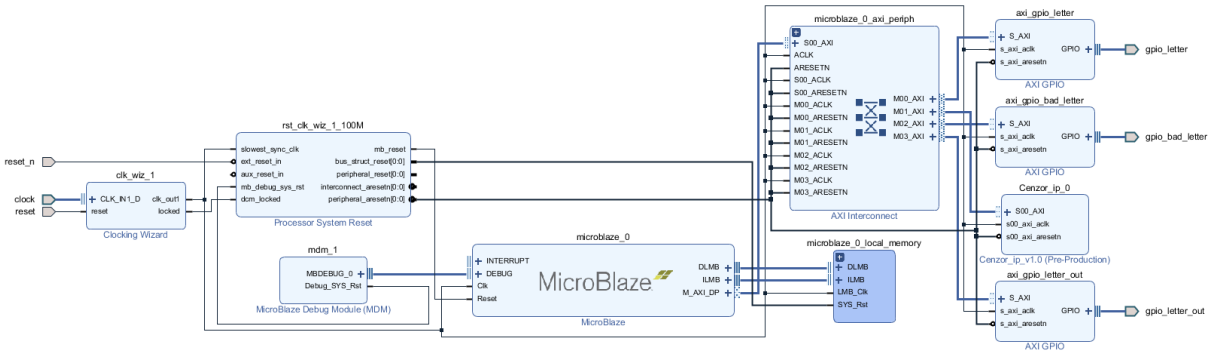
Przedstawiony model został przetestowany jako RTL. Następnie na jego bazie powstał projekt z systemem microBlaze, umożliwiający przetestowanie logiki i sygnałów potrzebnymi z poziomu języka C. porty bloku Cenzor zostały podłączone do rejestrów peryferyjnych AXI w sposób pokazany w tabeli 2.1. Ostatecznie powstał projekt zawierający stworzony blok IP wraz z systemem Zynq.



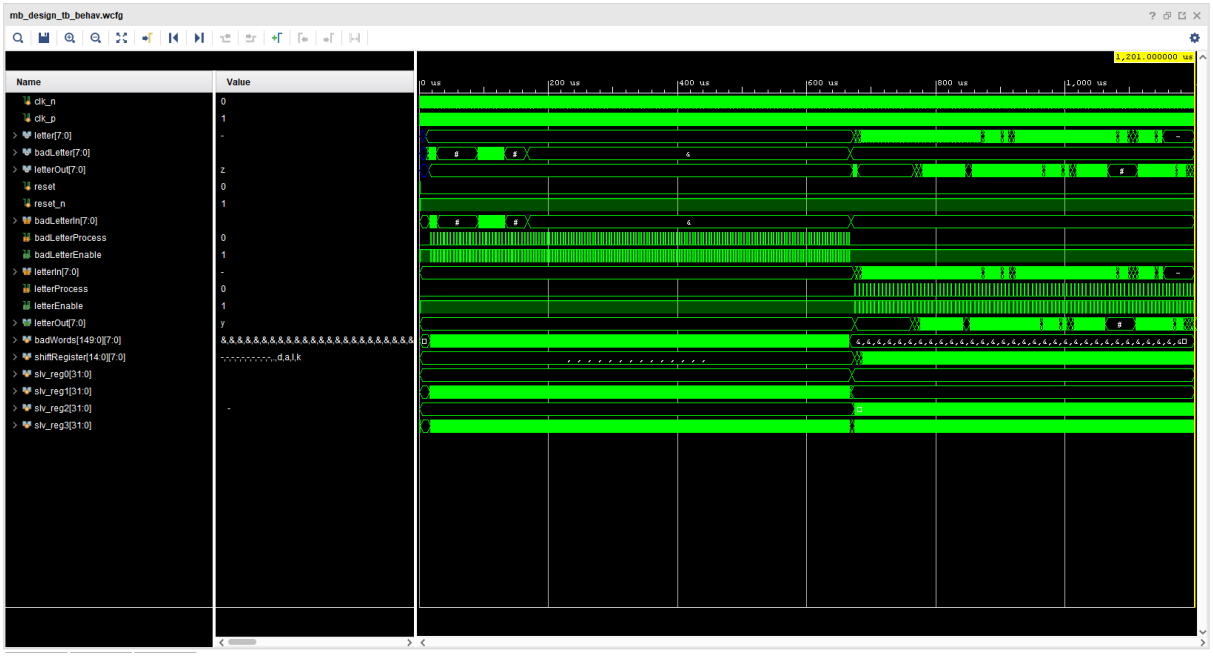
Rysunek 2.3: Przybliżona struktura wewnętrzna Cenzora.

Cenzor port	AXI mapped register	Direction	Port width
start	slv_reg0(0)	input	1
badLetterIn	slv_reg1(7:0)	input	8
badLetterProcess	slv_reg1(8)	input	1
badLetterEnable	slv_reg3(0)	output	1
letterIn	slv_reg2(7:0)	input	8
letterProcess	slv_reg2(8)	input	1
letterEnable	slv_reg3(1)	output	1
letterOut	slv_reg3(9:2)	output	8

Tabela 2.1: Przypisanie portów bloku Cenzor do rejestrów AXI.



Rysunek 2.4: Układ połączeń wykorzystując moduł MicroBlaze.



Rysunek 2.5: Obraz symulacji końcowej z poziomu MicroBlaze.

Z poziomu symulacji zauważyć możemy zasadniczo 2 procesy. Do około 670us, następuje proces wprowadzania liter do bufora badLetterIn, odpowiadających słowom do ocenzurowania. Litery wprowadzane są w formie bajtów, ze względu na kodowanie ASCI. Sytuacja wygląda podobnie w drugim procesie. Do bufora wejściowego LetterIn wchodzą kolejne litery słów tekstu do ocenzurowania i na bieżąco porównywane są ze słowami zabronionymi. Na początku tego procesu możemy zauważyć, latencje wynikającą z konieczności wprowadzenia do rejestru przesuwnego minimalnej ilości znaków. Opóźnienie obecne będzie również na końcu, wiąże się to z długością bufora, który musi zostać opróżniony.

3. Aplikacja

Do przygotowanego hardware'u została przygotowana aplikacja, potrzebna do sterowania, wpisywania słów do cenzorowania, wpisywania tekstu. Całość komunikacji z urządzeniem odbywa się poprzez port COM.

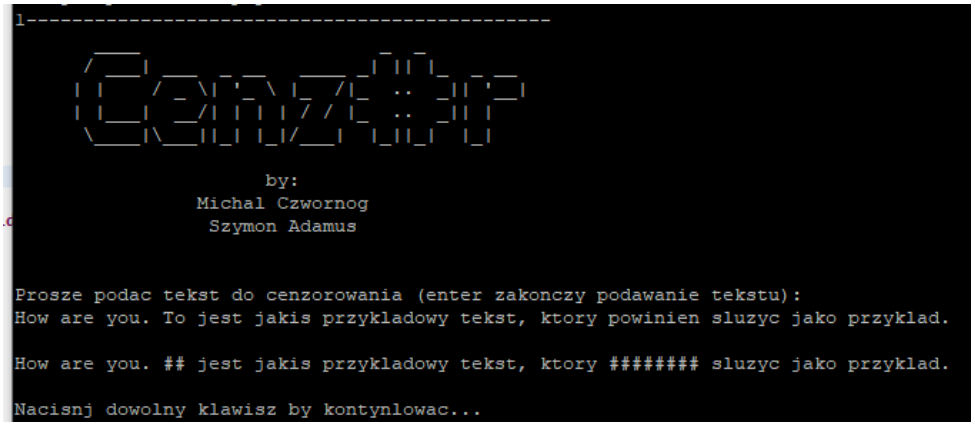
W aplikacji zostało stworzone proste menu, pokazane na rysunku 3.1. W menu możemy wybrać jedną z dostępnych opcji.

– Cenzoruj tekst - opcja ta pozwala użytkownikowi na wprowadzanie tekstu, którego wpisywanie zakańcza się znakiem końca linij. Następnie tekst ten jest cenzorowany i wyświetlany poniżej. Jak wygląda wejście i wyjście pokazano na rysunku 3.2.

– Pokaz slowa do cenzorowania - opcja ma na celu pokazanie użytkownikowi słowa jakie zostaną zamienione na "#" podczas cenzorowania tekstu. Przykładowe wyjście ukazuje rysunek 3.3.

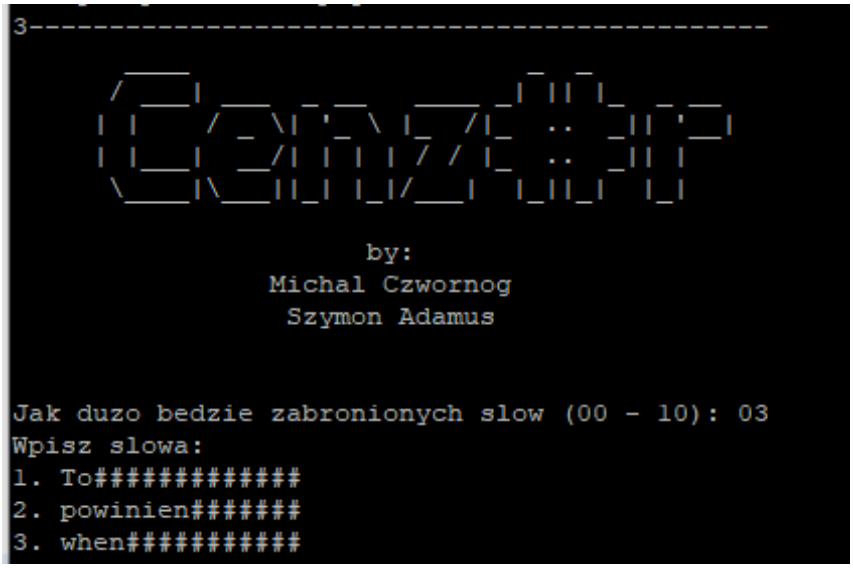
– Zmien slowa do cenzorowania - ostania opcja pozwala na zmienienie słów, które zostaną wykreślone z tekstu. Najpierw powinna zostać podana liczba słów (od 0 do 10), które będą podane. Następnie powinny zostać wpisane słowa o długości dokładnie 15 znaków. Jeżeli słowo ma mniej niż 15 znaków reszta słowa powinna zostać uzupełniona wildcard'em (czyli znakiem "#"). Przykład uzupełniania tych słów pokazano na rysunku 3.4.

Rysunek 3.1: Główne menu aplikacji.



Rysunek 3.2: Okno aplikacji podczas cenzorowania tekstu.

Rysunek 3.3: Okno aplikacji podczas wyboru drugiej opcji.



Rysunek 3.4: Zmiana słów do cenzorowania w aplikacji.

4. Podsumowanie

Założenia projektowe zostały zrealizowane. W kolejnych krokach kolejno zostały utworzone, RTL dla modułu Word_Comparator, na podstawie którego oparliśmy działanie modułu Cenzor RTL. Dla każdego z modułów wykonane zostały symulacje weryfikacyjne. Kolejnym krokiem było stworzenie projektu modułu IP dla Cenzora i wykorzystanie go przy współpracy z systemem MicroBlaze. Poprawność działania również została potwierdzona symulacjami. Ostatnim krokiem było przygotowanie projektu współpracującego z układem Zynq i stworzenie aplikacji w środowisku SDK.