Systemy Dedykowane w Układach Programowalnych

Dokumentacja projektowa

Autorzy:
Krzysztof Półchłopek
Michał Tomacha

1. Założenia projektowe

Celem projektu było stworzenie modułu do cenzurowania słów w ciągu tekstowym. Zadaniem procesora dedykowanego jest wykrywanie słów wprowadzonych do filtra, ocenzurowanie ich oraz zwrócenie tekstu z ocenzurowanymi już słowami. Projekt został oparty o filtr Blooma oraz dwie funkcje skrótu (CRC, hash Pearsona).

Projekt składa się z sześciu modułów:

- moduł główny cenzora
- moduł filtru Blooma,
- moduł obliczania CRC słowa
- moduł obliczania hasha Pearsona słowa
- moduł kolejki FIFO dla słów
- moduł kolejki FIFO dla informacji o konieczności ocenzurowania słowa

2. Filtr Blooma

2.1. Zasada działania

Filtr Blooma ma na celu reprezentację danego zbioru elementów oraz określanie przynależności jakiegoś elementu do tego zbioru. Filtr składa się z tablicy o określonej długości w której zapisane zostają określone informacje z funkcji haszujących. W zależności od ilości funkcji haszujących ustawiona zostaje odpowiednia ilość bitów. Sprawdzenie czy element **może** znajdować się w filtrze polega na porównaniu wartości pod indeksami z wartością 1. Jeżeli pod którymkolwiek z tych indeksów nie ma wartości 1, możemy uznać, że w tym filtrze nie znajduje się ten element.

W przypadku naszej implementacji: w rejestrze filtru ustawiana jest wartość 1 na bitach o indeksach wyników algorytmów haszujących.

Przykład:
Element 1 (CRC = 10, hasz Pearsona = 4)

Stan rejestru po wpisaniu nowych wartości:

Indeks	1	2	3	4	5	6	7	8	9	10
Wartość	0	0	0	1	0	0	0	0	0	1

2.2. Przypadek False-Positive

Ponieważ do filtru Blooma zazwyczaj wpisuje się więcej niż jeden element, może dojść do przypadku, że element, który domyślnie nie znajdował się w filtrze, po porównaniu jego haszy z filtrem zwróci wynik pozytywny. Dzieje się tak ponieważ filtr ten zwróci wynik pozytywny, jeżeli tylko elementy rejestru pod indeksami funkcji haszujących zwrócą wartości 1.

Przykład:
Element 1 (CRC = 10, hasz Pearsona = 4)
Element 2 (CRC = 7, hasz Pearsona = 6)

Stan rejestru po wpisaniu nowych wartości:

Indeks	1	2	3	4	5	6	7	8	9	10
Wartość	0	0	0	1	0	1	1	0	0	1

Element porównywany (CRC = 4, hasz Pearsona = 7)

Wynik tego porównania zwróci, że element ten znajduje się w filtrze pomimo tego, że nie był on wpisywany. Problem ten jest niemożliwy do rozwiązania jednak istnieją metody pozwalające na zmniejszenie częstotliwości występowania tego przypadku. Jedną z nich jest zwiększenie długości rejestru filtru.

W tym projekcie minimalizacja tego problemu została przeprowadzona w następujący sposób:

- długość rejestru filtru została ustawiona na 512 bitów,
- rejestr został podzielony na dwie równe części (każda 256 bitów) w których przechowywane są wyniki z poszczególnych funkcji haszujących

3. Działanie projektu

3.1. Moduł censora

Moduł z każdym taktem zegara pobiera wejściowy znak i wylicza na jego podstawie CRC i hasz Pearsona do momentu otrzymania znaku spacji, wtedy w zależności od linii sterującej bloom_write wpisuje wyliczone wartości haszy do filtru blooma (bloom_write = 1) lub porównuje obliczone wartości z filtrem (bloom_write = 0). Po porównaniu do wewnętrznego fifo wpisywany jest jego wynik od którego zależy czy słowo na wyjściu będzie zastąpione ciągiem znaków * odpowiadających jego długości. Dodatkowo w module znajduje się drugie fifo, które ma za zadanie przechowywanie wejściowych znaków do momentu sprawdzenia czy wejściowe słowo powinno być cenzurowane.

Moduł do poprawnego działania potrzebuje wejściowego ciągu znaków z którego będzie mógł odczytywać z każdym taktem zegara, więc w celu połączenia go z mikrokontrolerem zostało wykorzystane FIFO AXI-Stream.

Dzięki zastosowaniu haszowania podczas wpisywania słowa do filtru blooma jest nieograniczona, jednak w przypadku wpisywania słowa do porównania ograniczeniem jest długość bufora FIFO (maksymalnie 64 znaki). Dodatkowym ograniczeniem jest wielkość liter. Algorytmy haszujące rozróżniają wielkość liter przez co to samo słowo może mieć różne hasze w zależności od tego czy zaczyna się ono od dużej litery. Ponadto układ bierze pod uwagę znaki interpunkcyjne. Jeśli słowo kończy się dowolnym znakiem interpunkcyjnym, to hasze również zostaną policzone dla tych znaków. Oba te problemy da się rozwiązać kodem mikrokontrolera poprzez zamianę wszystkich liter na małe oraz usuwaniem określonych znaków interpunkcyjnych.

3.2. Program mikrokontrolera

Działanie programu rozpoczyna się od inicjalizacji UART oraz FIFO AXI-Stream. Następnie inicjalizowany jest moduł cenzora. Od tego momentu w konsoli wyświetla się prośba o wpisanie słów do ocenzurowania. Po wpisaniu przez użytkownika pojawia się kolejny komunikat z prośbą o wprowadzenie tekstu do ocenzurowania. Cały tekst jest następnie umieszczany w FIFO AXI wraz z dodatkowymi informacjami dla modułu cenzora oraz zostaje przesłany do układu FPGA. Następnie układ ten cenzuruje tekst oraz zwraca go na odbiorcze FIFO AXI. Finalnie tekst zostaje wypisany na port szeregowy.