AKADEMIA GÓRNICZO-HUTNICZA

Raport z projektu

„Fibonacci search technique”

z przedmiotu

Systemy Dedykowane w Układach Programowalnych

Elektronika i Telekomunikacja

Systemy Wbudowane, rok I

Jakub Górnisiewicz

Przemysław Kapała

22.06.2023

Spis treści

1. Opis teoretyczny 3

Zasada działania algorytmu 3

Zastosowanie 3

„Search” 3

„Minimum” 3

2. Implementacja 3

Python 3

System Verilog 4

3. Weryfikacja działania 4

„Search” 4

„Minimum” 5

4. Podsumowanie 5

1. Opis teoretyczny

Technika wyszukiwania Fibonacciego to metoda przeszukiwania posortowanej tablicy przy użyciu algorytmu dziel i zwyciężaj, który zawęża możliwe lokalizacje przy pomocy ciągu Fibonacciego.
W porównaniu do wyszukiwania binarnego, gdzie posortowana tablica jest dzielona na dwie równoliczne części, z których jedna jest badana dalej, wyszukiwanie Fibonacciego dzieli tablicę na dwie części o rozmiarach będących kolejnymi liczbami Fibonacciego.

Zasada działania algorytmu

Cały pseudo algorytm metody wyszukiwania Fibonacciego znajduję się pod linkiem: https://en.wikipedia.org/wiki/Fibonacci_search_technique

Zastosowanie

W projekcie zaimplementowano dwie funkcjonalności: „Search" oraz „Minimum".

„Search"

Implementacja ta ma za zadanie zwrócić indeks, pod którym znajduje się szukana wartość. Użytkownik podaje na wejściu modułu posortowaną w kolejności rosnącej tablice liczb całkowitych (rozmiar tablicy musi być znany) i szukaną wartość a algorytm zwraca indeks pod którym wyraz występuje w tablicy. W przypadku braku występowania zadanego elementu algorytm nas o tym informuje.

„Minimum"

Implementacja zwraca argument funkcji, dla którego występuje minimum lokalne funkcji (maksymalnie jedno ekstremum w przedziale). Użytkownik podaje współczynniki wielomianu, przedział o zadanej szerokości, który chcemy przeszukać oraz dokładność przybliżenia wyniku. W celu usprawnienia algorytmu można zamiast wykorzystywać dzielenie kolejnych wyrazów ciągu Fibonacciego zastosować wyznaczone wcześniej dwie wartości (skalujące podprzedziały) dające wystarczająco dokładny wynik.

2. Implementacja

Python

Kod źródłowy w języku Python został podzielony na pięć plików:

- fibonacci_serach.py
- tb_fibonacci_serach.py
- fibonacci_minimum.py
- tb_fibonacci_minimum.py
- data_array_generation.py

Pierwsze dwa określają funkcjonalność implementacji „Search" oraz weryfikują jej działanie z wykorzystaniem Unitestów.
Kolejne dwa implementacje „Minimum" wraz z weryfikacją.

Ostatni plik generuje tablicę posortowanych elementów wykorzystywanych w metodzie „Serach”.

System Verilog

Kod źródłowy w języku System Verilog został podzielony na sześć plików oraz dodatkową funkcję:

- fibonacci.sv
- tb_fibonacci.sv
- fibonacci_search.sv
- tb_fibonacci_search.sv
- fibonacci_minimum.sv
- tb_fibonacci_minimum.sv
- funkcja f

Moduł fibonacci służy do wyliczenia kolejnych elementów ciągu Fibonacciego.
Moduł fibonacci_search odpowiada za implementacje „Search”.
Moduł fibonnaci_minimum odpowiada za implementacje „Minimum”.
Funkcja f wykorzystywana jest w fibonnaci_minimum oraz służy do wyznaczania wartości funkcji dla zadanego argumentu.

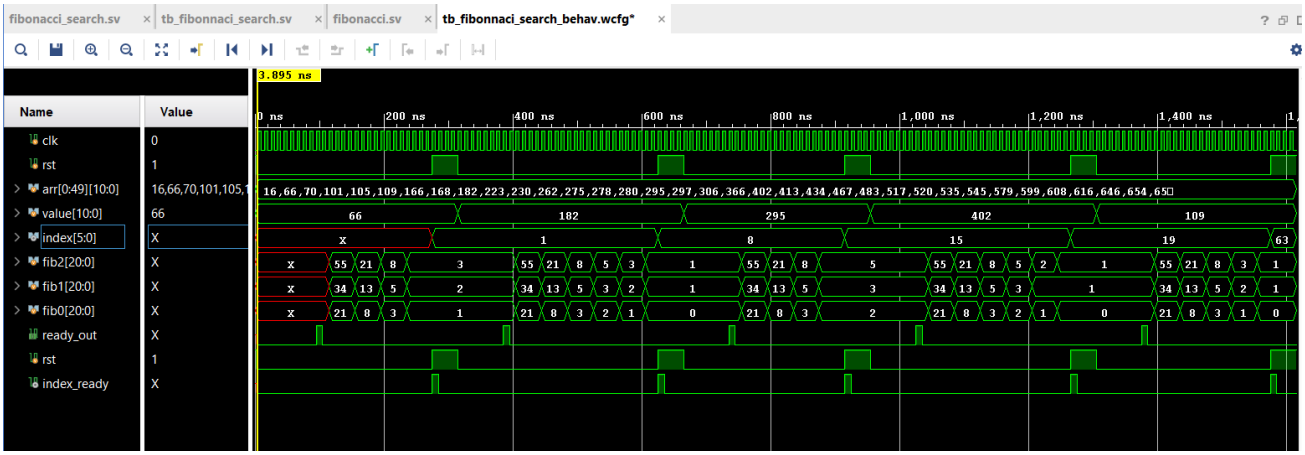
*function_to_fibonacci - służy do weryfikacji implementacji oraz testowania „funkcji f”. Nie jest wymagana do poprawnego działania projektu.

3. Weryfikacja działania

Syntezowalny jest tylko algorytm „Search”, jest on zgodny z głównym celem projektu „Fibonacci Serach Technique” opisanym na Wikipedii.

„Search”

Kod zaimplementowano w dwóch wersjach: maszyna stanów (switch case) oraz z wykorzystaniem list warunkowych (if).
Wersja „if” jest szybsza od wersji „switch case”.

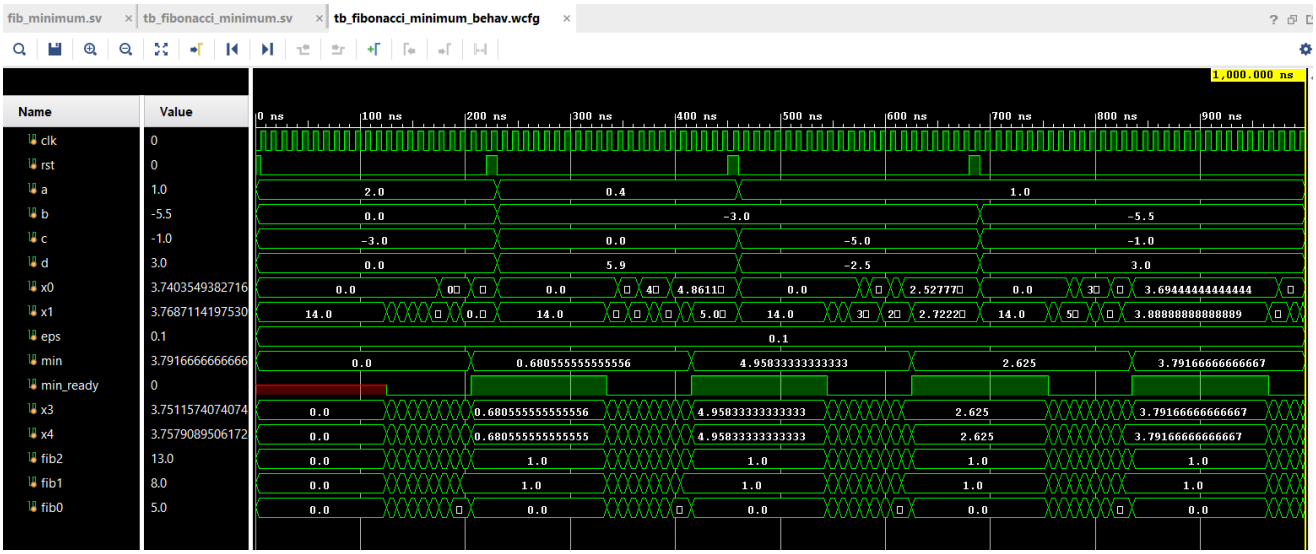


Rysunek 1. Przebiegi czasowe Search

```
# run 1000ns
PASS
Index:  1
wartosc:   66
PASS
Index:  8
wartosc:  182
PASS
Index: 15
wartosc:  295
INFO: [USF-XSim-96] XSim completed. Design snapshot 'tb_fibonnaci_search_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:05 ; elapsed = 00:00:06 . Memory (MB): peak = 922.145 ; gain = 0.000
run 20 us
PASS
Index: 19
wartosc:  402
PASS
Index: 63
wartosc:  109
```

Rysunek 2. Weryfikacja w TCL Search

„Minimum”



Rysunek 3. Przebiegi czasowe Minimum

Szukane minima powinny wynosić: 0.707, 5, 2.633, 3.755.
Obliczone minima: 0.681, 4.958, 2.625, 3.792.

4. Podsumowanie

Mimo początkowych problemów ze składnią języka System Verilog udało się Nam finalnie ukończyć projekt. Stawiane przez Nas założenia zostały zrealizowane. Podczas wykonywania projektu nauczyliśmy się myśleć bardziej kreatywnie oraz uwzględniać problemy, na które nie napotykaliśmy w innych językach. Dodatkowym aspektem było poznanie nowego algorytmu do przeszukiwania tablic, który jest znacznie wydajniejszy od klasycznego.

Zdajemy sobie sprawę iż zaimplementowane przez Nas kody nie są doskonałe oraz pozostawiają bardzo duże pole do poprawy. Dołożyliśmy wszelakich starań, aby projekt spełniał kryterium projektu na studiach magisterskich mimo, że programowanie FPGA nie jest w kręgu Naszych zainteresowań.