Technika Cyfrowa – Sprawozdanie z ćwiczenia 4

**Praktyczna implementacja projektów cyfrowych w rzeczywistym programowalnym układzie scalonym FPGA**

**Autorzy: Robert Raniszewski, Kacper Feliks, Paweł Czajczyk, Mateusz Pawliczek**

# 1. Treść zadania

Korzystając z programu Quartus firmy Altera (Intel) (www.altera.com), należy w dowolnym wybranym układzie scalonym FPGA stworzyć działający system sterujący, realizujący bardzo prosty wybór z menu. Mamy do dyspozycji dwa wyświetlacze siedmiosegmentowe (lewy i prawy) oraz dwa przyciski (lewy i prawy). Wciśnięcie lewego przycisku powoduje zwiększanie o jeden wartości wyświetlanej na lewym wyświetlaczu. Po osiągnięciu wartości 9, wartość ta zmienia się na 0 przy ponownym wciśnięciu lewego przycisku. Po już ustawieniu lewym przyciskiem żądanej wartości na lewym wyświetlaczu, wciśnięcie prawego przycisku powinno spowodować zapamiętanie wartości z lewego wyświetlacza na wyświetlaczu prawym, co świadczy o dokonanym wyborze wartości z menu.

Po uruchomieniu systemu, obydwa wyświetlacze powinny pokazywać wartość zero. Całość powinna działać zgodnie z filmem z poniższego linku:

<https://home.agh.edu.pl/~dlugopol/tc2025/fpga-menu.mp4>

# 2. Układy FPGA

FPGA to programowalne układy cyfrowe, które umożliwiają tworzenie własnych struktur logicznych bez potrzeby projektowania dedykowanego układu scalonego.

Układ FPGA składa się z tysięcy konfigurowalnych bloków logicznych, układów dodających i przerzutników typu D. Programowanie FPGA odbywa się najczęściej w językach opisu sprzętu, takich jak VHDL lub Verilog.

W przeciwieństwie do tradycyjnych mikroprocesorów czy mikrokontrolerów, FPGA nie wykonują kodu sekwencyjnie i posiadają większą swobodę działania w architekturze.

# 3. Praktyczne zastosowania FPGA

Układy FPGA znajdują szerokie zastosowanie w wielu dziedzinach, m.in.:

* **Elektronika przemysłowa:** sterowniki maszyn,
* **Telekomunikacja:** kodowanie i dekodowanie sygnałów, szybkie przetwarzanie strumieni danych,
* **Motoryzacja:** systemy wspomagania kierowcy (ADAS), czujniki i układy przetwarzające sygnały w czasie rzeczywistym,
* **Medycyna:** obrazowanie medyczne, szybkie przetwarzanie sygnałów z czujników,
* **Obrona i lotnictwo:** systemy radarowe, komunikacja szyfrowana, analiza sygnałów w czasie rzeczywistym,

# 4. Rozwiązanie zadania

Do rozwiązania zadania wykorzystaliśmy układ FPGA Altera FLEX EPF10K70RC240-4 oraz program Altera Quartus II ver. 9.0.

## 4.1 Wersja podstawowa

System został zrealizowany w środowisku Quartus na układzie FPGA i wykorzystuje dwa przyciski oraz dwa wyświetlacze siedmiosegmentowe.

Po uruchomieniu oba wyświetlacze pokazują 0. Wciśnięcie **lewego przycisku** zwiększa wartość na **lewym wyświetlaczu** (od 0 do 9, potem od nowa). Wciśnięcie **prawego przycisku** powoduje **przepisanie** tej wartości na **prawy wyświetlacz**, co oznacza zatwierdzenie wyboru.

## 4.2 Dodatkowe modyfikacje

Do podstawowego działania systemu dodaliśmy kilka funkcji:

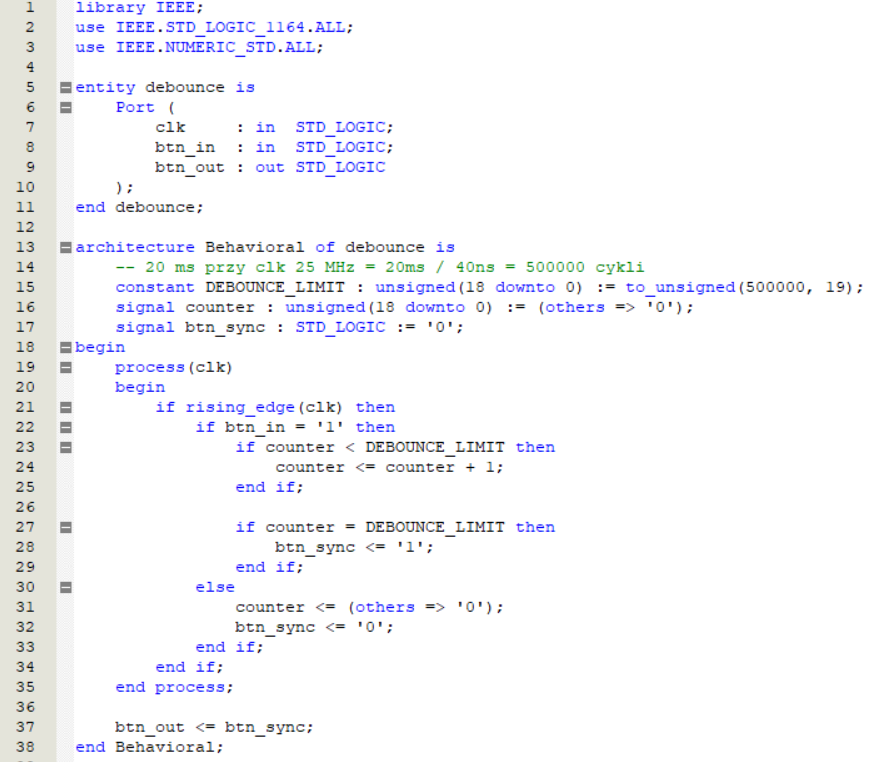
1. **Auto-repeat** – długie przytrzymanie przycisku powoduje automatyczne, powtarzające się zwiększanie (lewy przycisk) lub zmniejszanie (prawy przycisk) wartości. Częstotliwość powtórzeń rośnie z czasem – po około 2 sekundach przytrzymania interwał zmniejsza się aż do 0,5 sekundy.
2. **Odejmowanie przy prawym przycisku** – przyciśnięcie **prawego przycisku** powoduje zmniejszenie wartości na lewym wyświetlaczu. Jeśli osiągnie 0, przechodzi na 9 (cyklicznie w dół).
3. **Zapis obu przyciskami** – jeśli oba przyciski zostaną wciśnięte jednocześnie, wartość z lewego wyświetlacza zostaje zapisana na prawy, niezależnie od wcześniejszego działania.

Wszystkie przyciski są objęte **mechanizmem debouncingu**, aby wyeliminować fałszywe przełączenia wywołane drganiami styków.

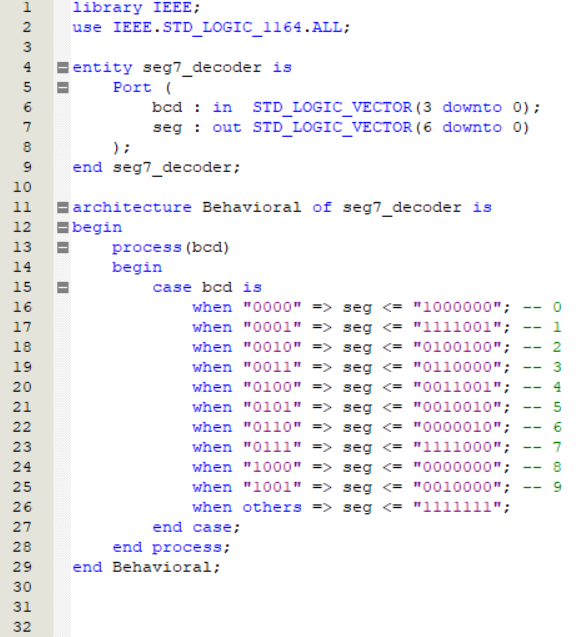
# 5. Kod projektu

W osobnych plikach zdefiniowaliśmy obsługę debouncingu oraz konwersję liczb na wyświetlacz siedmiosegmentowy.

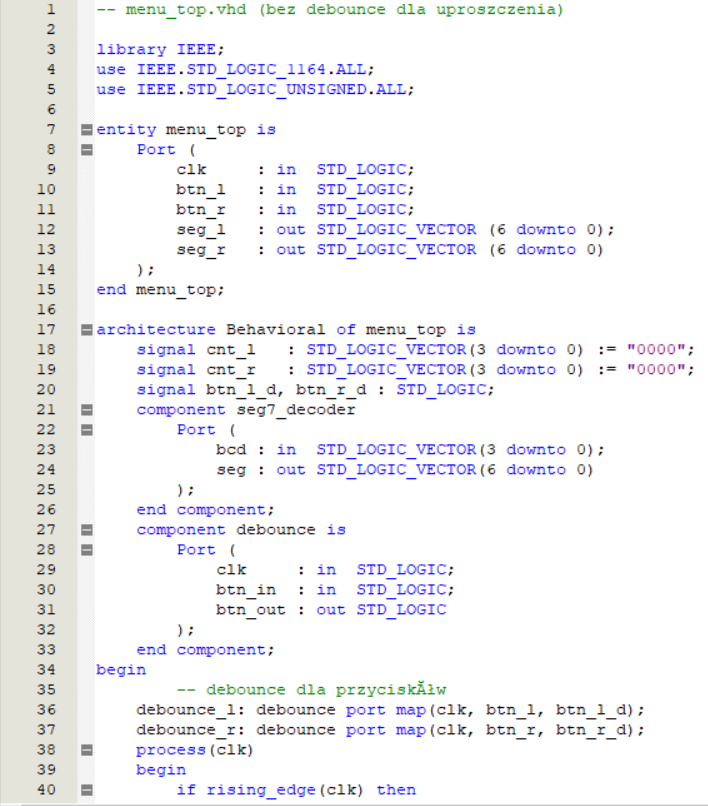
Debouncing:

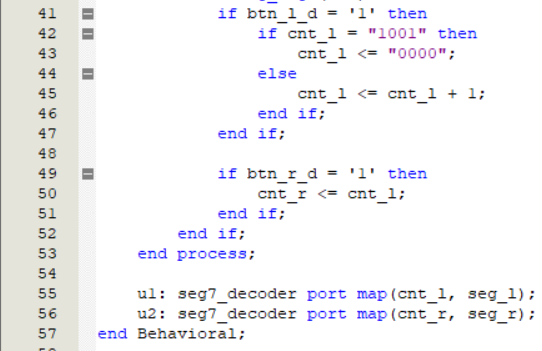


seg7\_decoder:

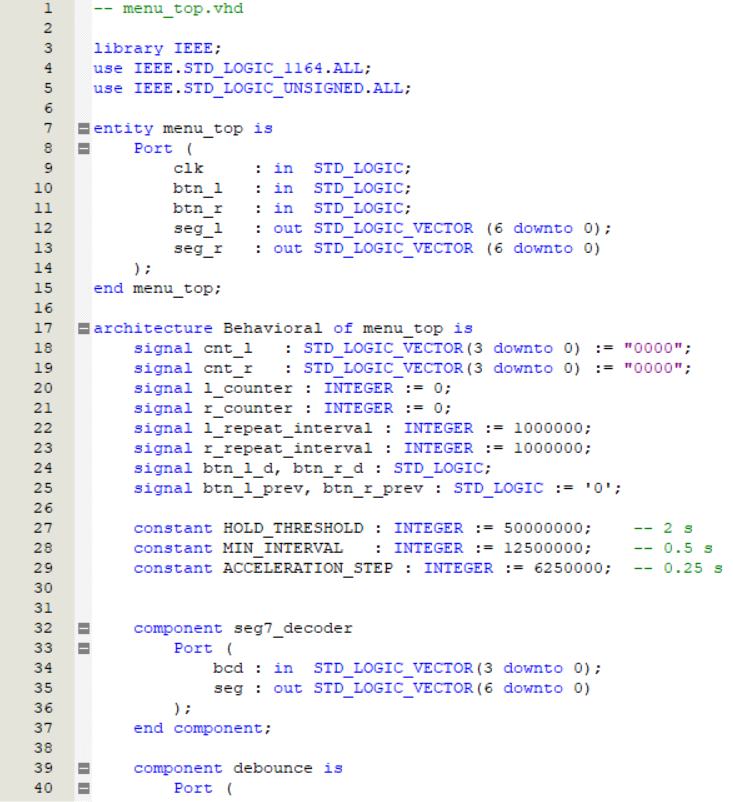


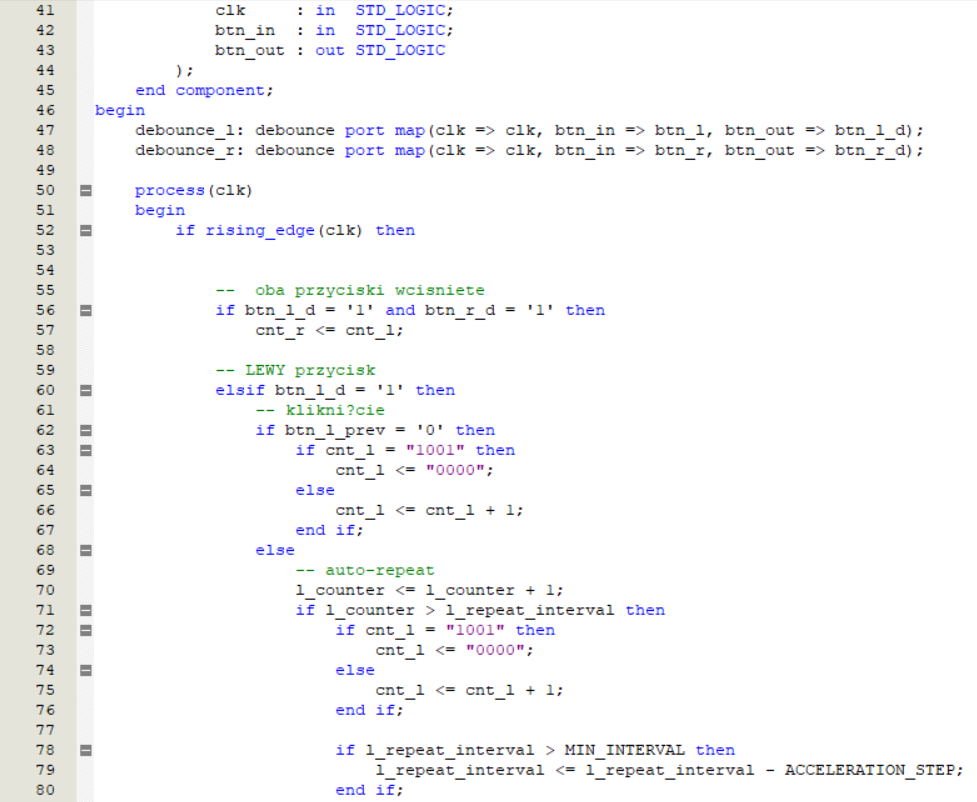
Wersja podstawowa projektu:

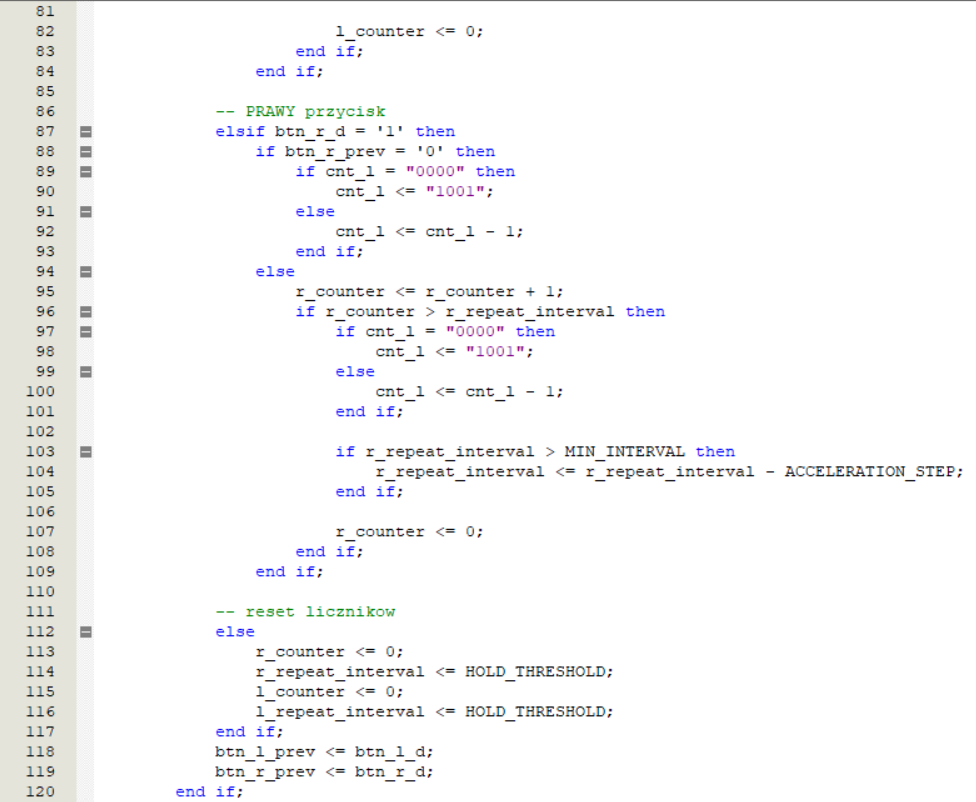


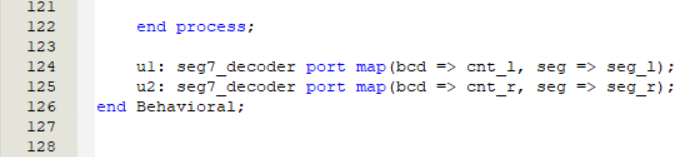


Wersja zmodyfikowana:









# 6. Przypisanie pinów

