Tytuł: **Obsługa głosowa led basys3**

Autorzy: **Mateusz Gibas (MG), Kacper Ferdek (KF)**

Ostatnia modyfikacja: 31.08.2024

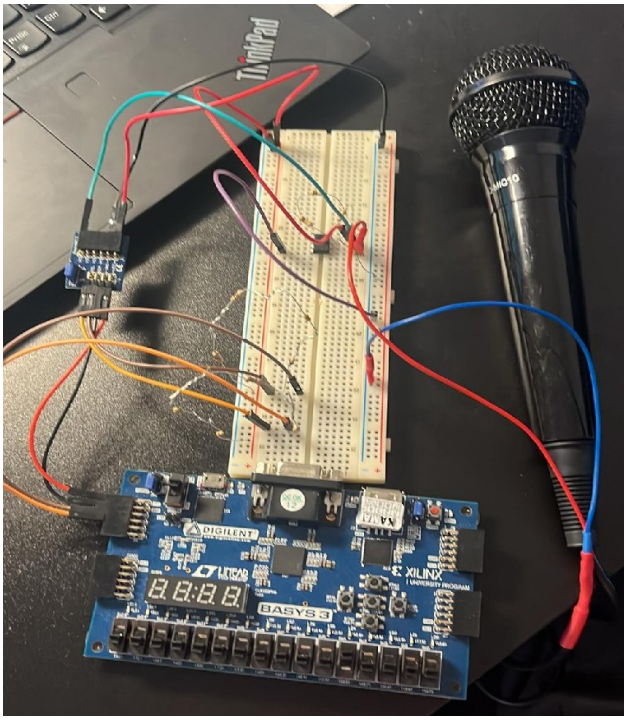
Spis treści

1. Repozytorium git

Adres repozytorium GITa :https://github.com/Ferdziu10/Simple-speech-recognisition

2. Wstęp

Do płytki basys3 jest podpięte adc, do którego podpięty jest mikrofon. Płytka po usłyszeniu jednej z dwóch komend będzie zapalała lub gasiła diodę. Przełączenie switcha będzie skutkowało brakiem możliwości zmiany stanu diody. Po prawej stronie znajduje się zdjęcie rzeczywistego układu składającego się z mikrofonu, płytki fpga basys3, przetwornniku ADC AD7991 Pmod, rezystorów, płyki stykowej oraz oczywiście kabelków łączących w odpowiedni sposób wszystkie elementy.

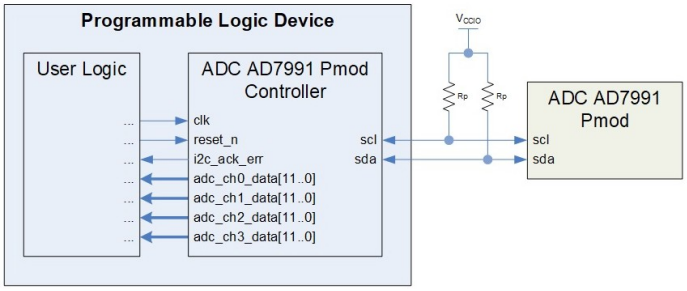


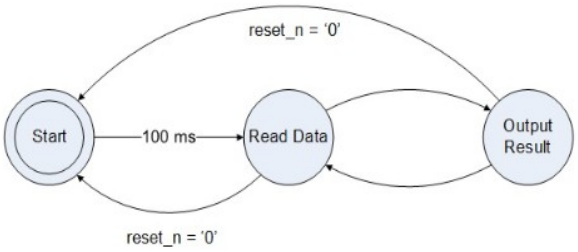
3. Specyfikacja

3.1. Opis ogólny algorytmu

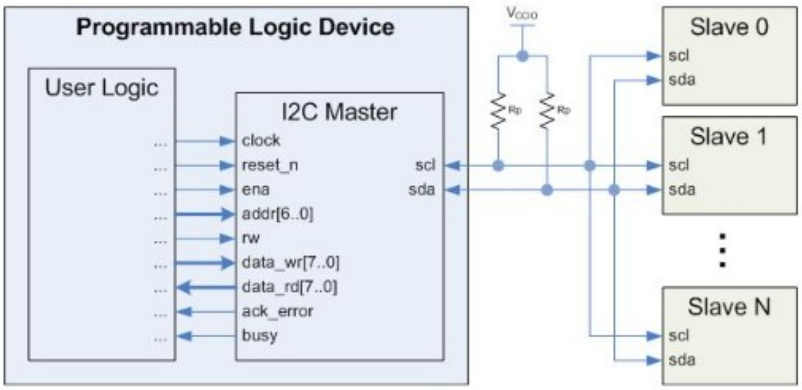
Działanie programu składa się z 4 głównych etapów:

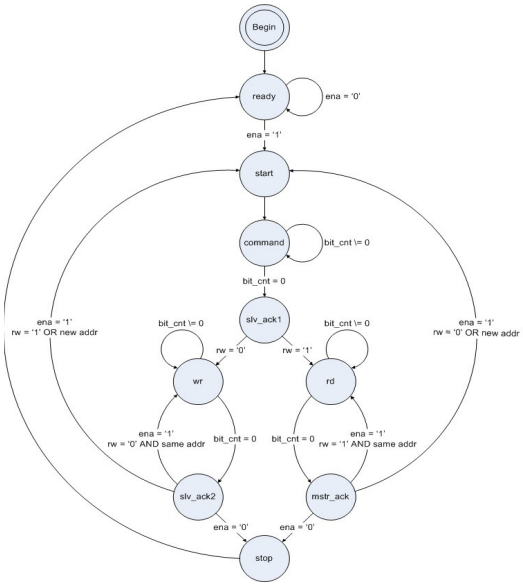
1. Grupy modułów kontrolujących adc i przesył danych do dalszej części programu, poniżej znajduje się schemat połączenia płytki z adc i modułu kontrolującego jego pracę, posiada on strukturę maszyny stanów, której uproszczony schemat także znajduje się poniżej (oryginalna wersja z resetem sterowanym stanem niskim), moduł ten służy tylko do obsługi i konfiguracji właściwego modułu i2c_master, który faktycznie zajmuje się komunikacją z tzw. slave'ami (w naszym przypadku jednym, którym jest adc).



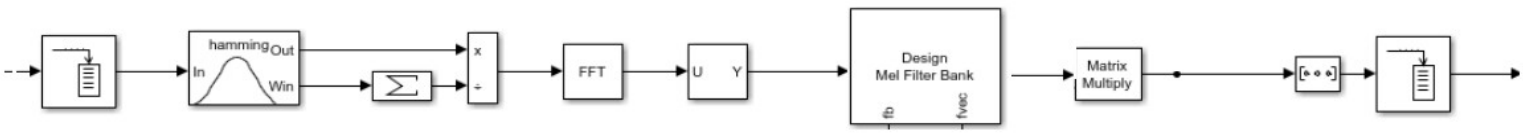


Poniżej znajduje się schemat działania układu, a także diagram maszyny stanów, tak samo jak wprzypadku Pmod Controller

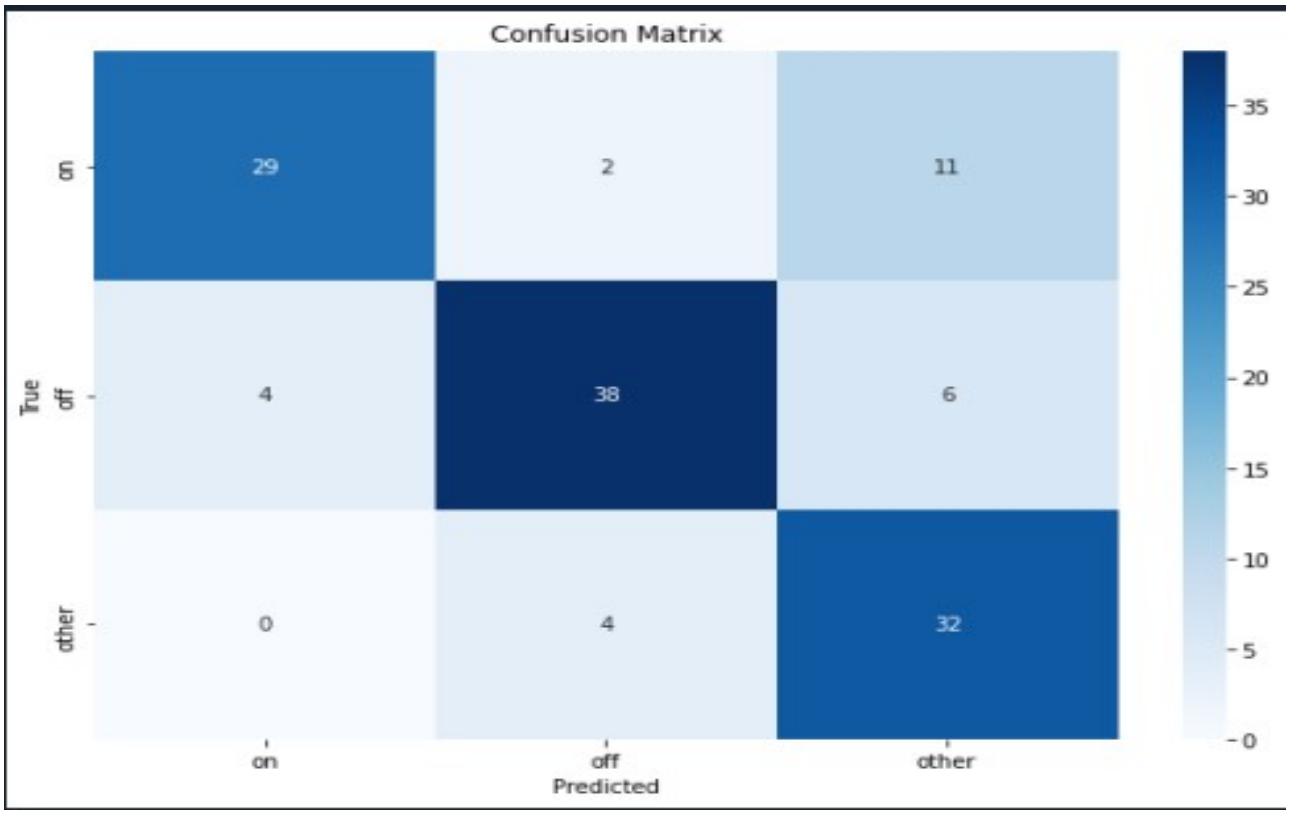




2. Odpowiedniej obróbki sygnału audio otrzymanych z adc w celu przygotowania ich do przesłania do sieci neuronowej. Schemat blokowy, użytego do tego algorytmu znajduje się poniżej



3. Rozpoznania konkretnych komend przez sieć neuronową na podstawie dostarczonych danych po obróbce audio. Dane wejściowe przechodzą przez kolejne warstwy sieci, w każdej z nich są transformowane w sposób, który stopniowo umożliwia sieci ustalić jakie słowo zostało wypowiedziane do mikrofonu. Poniżej przedstawiono tabele, która pokazuje jak często sieć faktycznie rozpoznaje słowa w idealnych warunkach.



4. Ostatnim etapem jest prosta logika, która ustawia stan diody w zależności od otrzymanej odpowiedzi z sieci neuronowej, a także obecnej pozycji switch'a

3.2. Tabela zdarzeń

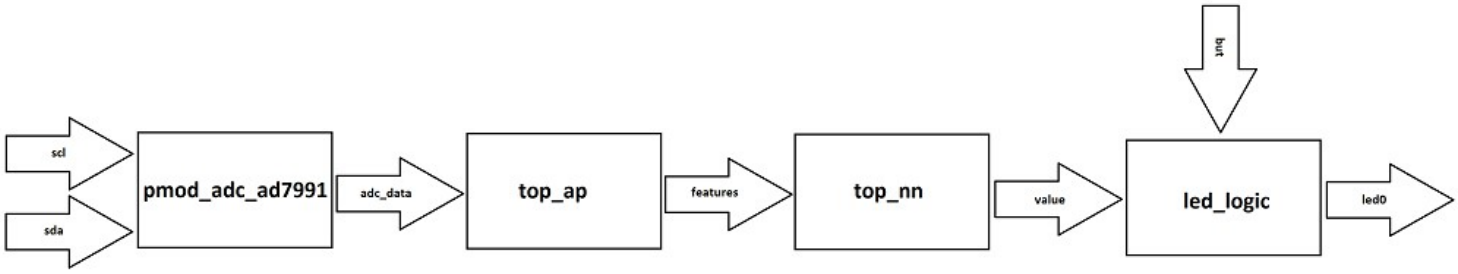
Opis zdarzeń występujących podczas działania programu/urządzenia, zarówno zewnętrznych (interakcje z użytkownikiem), jak i wewnętrznych (specyficzne stany w algorytmie). Zdarzenia podzielone są na kategorie dotyczący różnych stanów działania programu. Kategorie powinny odpowiadać stanom ze schematu z pkt. 2.1.

Zdarzenie	**Kategoria**	**Reakcja systemu**
Rozpoznanie komendy „off”	Rozpoznawanie mowy	Zgaszenie diody
Rozpoznanie komendy „on”	Rozpoznawanie mowy	Zaświecenie diody
Rozpoznanie komendy „other”	Rozpoznawanie mowy	Brak zmian stany diody
Przełączenie switcha na 1	Kontrola diody	*Odblokowanie możliwości zmiany stanu diody*
Przełączenie switcha na 0	Kontrola diody	Zablokowanie *możliwości zmiany stanu diody*

4. Architektura

4.1. Moduł: top

4.1.1. Schemat blokowy



4.1.2. Porty

a) input

nazwa portu	**opis**
but	Wejście na switch sterujący możliwością zmiany stanu diody
scl(**in**out)	wyznacza „rytm” w jakim nadawane są kolejne bity adresów
sda(**in**out)	transmisja właściwych informacji pomiędzy współpracującymi układami (w naszym przypadku płytka i adc)

b) output

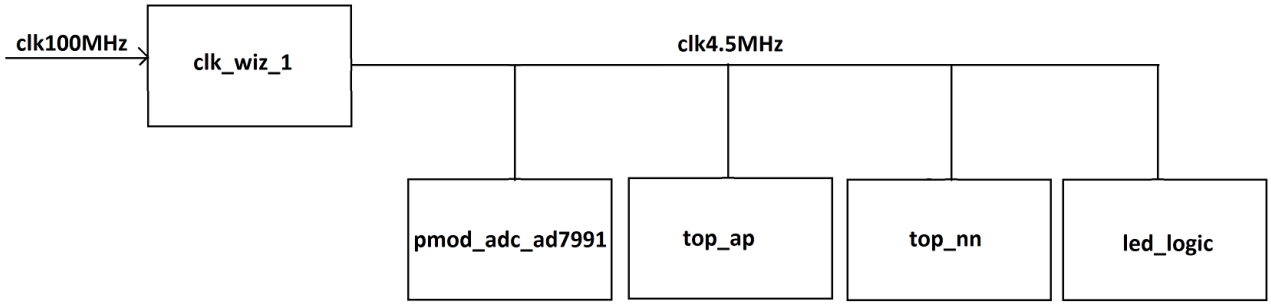
nazwa portu	**opis**
led0	wyjście sterujące stanem diody

4.1.3.
Interfejsy

Ten projekt nie zawiera interfejsów, ponieważ przez całą długość projektu do dalszych modułów jest przesyłany zazwyczaj tylko jeden sygnał, który dodatkowo między modułami zmienia swoją szerokość.

4.2. Rozprowadzenie sygnału zegara

W wszystkich modułach używany jest zegar o tej samej częstotliwości wynoszącej 4.5 MHz. Został on wygenerowany dzięki generatorowi, który z kolei został wygenerowany przy użyciu IP core zwanego clocking wizard. W naszym przypadku otrzymuje on na wejściu sygnał zegarowy o częstotliwości 100 MHz (generator takiego sygnału zegarowego jest dostarczony na płytce basys3 fabrycznie), a na wyjściu podaje sygnał 4.5 MHz.

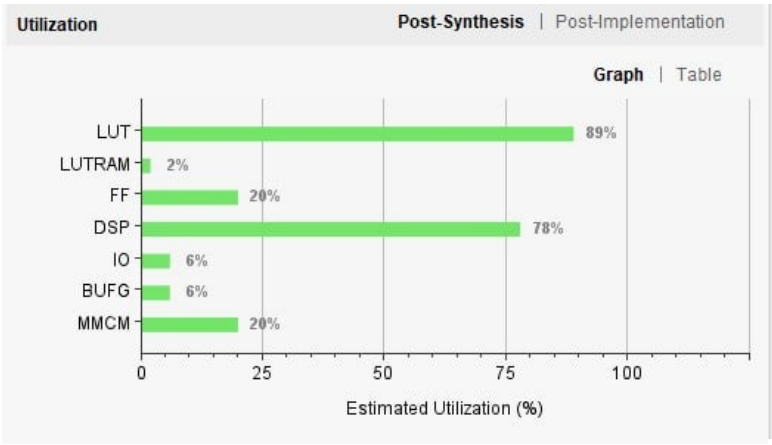


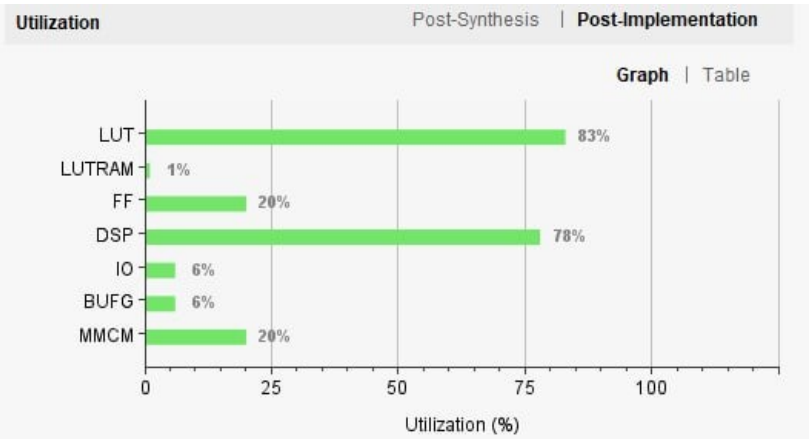
5. Implementacja

5.1. Lista zignorowanych ostrzeżeń Vivado.

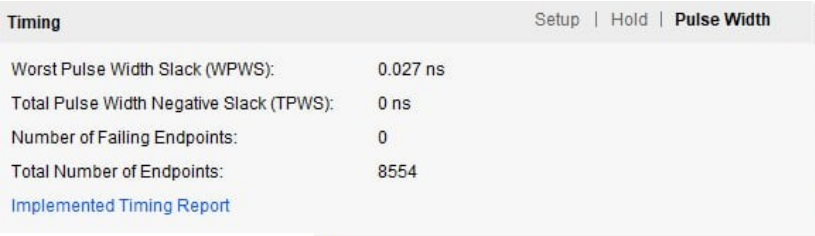
Identyfikator ostrzeżenia	Liczba wystąpień	Uzasadnienie
Synth 8-3848	1	Ostrzeżenie występujące w zapożyczonym module związane z tym ze wpisujemy danych do slave komponentów, a jedynie sczytujemy z nich dane
Synth 8-6014	4	Usunięcie podanych elementów skutkuje niepoprawnym działaniem modułu
Synth 8-7080	1	Pospolicie występujące ostrzeżenie nie wpływające na działanie układu
Synth 8-3323	1	Ostrzeżenie nieistotne, w implementacji DSP wystarczyło
Power 33-332	1	Po otworzeniu power reportu nie można go odnaleźć, tak jakby nie istniał
DRC DPIP-1	100	Ostrzeżenia związane z pipeliningiem, 1 część pochodzi z modułu zapożyczonego, 2 część z modułu mean_std, w którym zastosowanie częściowego pipeliningu nie usuwa błędu, a 3 część z modułu multiplier, można go usunąć stosując ip dostarczony w programie, aby to zrobić należy od komentować jego instancje w module mel_filter_bank i przenieść folder ip, znajdujący się w tym samym folderze co poprzednio wspomniany moduł do następującej ścieżki Simple-speech-recognition/fpga/build/ss_projject.gen/sources_1 warto zaznaczyć, że symulacja działa tylko jeżeli nie używa się ip
DRC DPOP-2	68	

5.2. Wykorzystanie zasobów





5.3. Marginesy czasowe



WNS	TNS	WHS	THS	WBSS	TPWS
22.469	0.000	0.013	0.000		0.000

6. Film.

Link do ściągnięcia filmu:

https://www.youtube.com/watch?v=afWAw01X5aQ

https://drive.google.com/file/d/1JsHOwn726z8mybDmwwTvAIakW7A-SqXD/view