XII KRAJOWA KONFERENCJA ELEKTRONIKI
Darłówko Wschodnie, 10 - 13.06. 2013

dr inż. Tomasz Marciniak
mgr inż. Radosław Weychan
mgr inż. Agnieszka Stankiewicz
prof. dr hab. inż. Adam Dąbrowski

Wydział Informatyki
Katedra Sterowania i Inżynierii Systemów
Pracownia Układów Elektronicznych i Przetwarzania Sygnałów
Politechnika Poznańska
ul. Piotrowo 3, 60-965 Poznań

Implementacja przetwarzania biometrycznego sygnału mowy w systemie z procesorem sygnałowym

Słowa kluczowe: biometria, przetwarzanie mowy, procesor sygnałowy, mieszane modele Gaussa, kwantyzacja wektorowa

STRESZCZENIE

Artykuł prezentuje analizę zagadnień związanych z implementacją przetwarzania biometrycznego sygnału mowy realizowanego z wykorzystaniem stałoprzecinkowego procesora sygnałowego. Do badań eksperymentalnych wykorzystano środowisko obliczeniowe Matlab oraz środowisko uruchomieniowe dla procesorów sygnałowych Code Composer Studio (CCS). Przetwarzanie sygnału mowy zrealizowano z wykorzystaniem modułu uruchomieniowego z procesorem TMS320C5515, przy czym celem działania systemu była identyfikacja mówcy. Przeanalizowano ograniczenia związane z działaniem algorytmu w systemie wbudowanym. Zademonstrowano techniki konwersji oprogramowania ze środowiska Matlab do CCS. Pokazano wpływ reprezentacji stałoprzecinkowych na skuteczność identyfikacji.

1. WPROWADZENIE

Współczesne mikrokontrolery posiadają wiele cech, które dotychczas były właściwościami charakterystycznymi procesorów sygnałowych. Przykładowo, aktualnie produkowane przez firmy Atmel czy Microchip mikrokontrolery posiadają architekturę harwardzką i sprzętowy układ mnożący, który potrafi dokonywać operacji mnożenia także na stałoprzecinkowych liczbach ułamkowych. Należy jednak pamiętać, że przetwarzanie sygnału mowy jest wspomagane poprzez dedykowane rozwiązania sprzętowe, dostępne tylko w procesorach sygnałowych. Przykładem takiego rozwiązania jest zwielokrotnianie jednostek wykonawczych czy adresowanie z rewersją bitów, a nawet akcelerator obliczania dyskretnej transformaty Fouriera technikami FFT (*fast Fourier transformation*) [1, 2]. Zaletą współczesnych procesorów sygnałowych jest też zoptymalizowane zużycie energii, pozwalające na pracę takich układów w systemach przenośnych, zasilanych za pomocą niewielkich akumulatorów. Najważniejsze parametry stosowanego przez autorów w badaniach procesora sygnałowego omówiono w punkcie 3.

Przetwarzanie sygnałów mowy w aplikacjach realizujących określone zadanie (np.

kompresja, odszumianie czy też rozpoznawanie mowy lub mówcy) składa się najczęściej z kilku etapów. Przykładem takiego kilkuetapowego przetwarzania mowy jest rozpoznawanie mówcy, które wymaga parametryzacji sygnału wejściowego (dzielenia sygnału na bloki, mnożenia przez funkcję okna, wyznaczenia współczynników predykcji liniowej lub mel-cepstralnych za pomocą FFT), a także modelowania (np. kwantyzacji wektorowej i tworzenia mieszanin gaussowskich, w skrócie GMM) a następnie porównywania z bazą wzorców [3].

2. TECHNIKI ROZPOZNAWANIA MÓWCY Z ZASTOSOWANIEM SYSTEMÓW WBUDOWANYCH

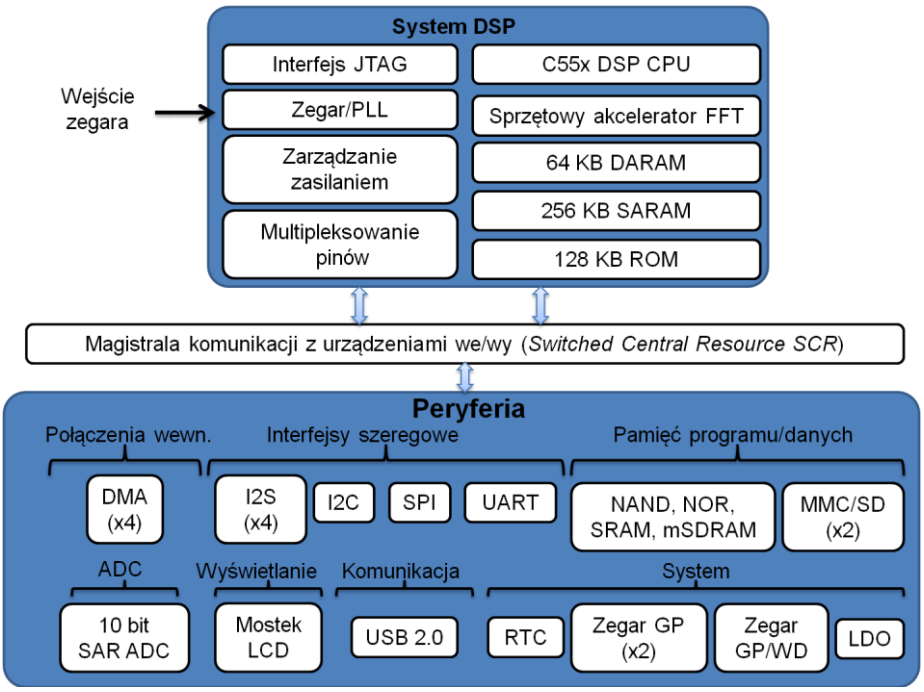
Automatyczne rozpoznawanie mówcy jest zagadnieniem, które jest badane od ponad 50 lat [4]. W bazie IEEE Xplore dla hasła „*speaker recognition*" znajduje się ponad 6400 publikacji, przy czym w ostatnich 3 latach pojawiło się ich aż 1300. Systemy rozpoznawania mówcy wykorzystują różne technologie: sieci neuronowe, modelowanie za pomocą GMM (*Gaussian mixture models*) czy maszyny wektorów nośnych SVM (*support vector machines*) [5].
Kluczowe elementy, na jakie zwracają uwagę autorzy publikacji dotyczących realizacji rozpoznawania mówcy w systemach wbudowanych, są następujące [6, 7, 8, 9]:

- jakość i odszumienie sygnału wejściowego
- wpływ rozdzielczości danych podczas przetwarzania
- wykorzystanie zasobów sprzętowo-programowych
- rozmiary pamięci systemu
- ułatwienia związane z implementacją programową np. dostępność bibliotek.

3. MODUŁ Z PROCESOREM SYGNAŁOWYM C5515

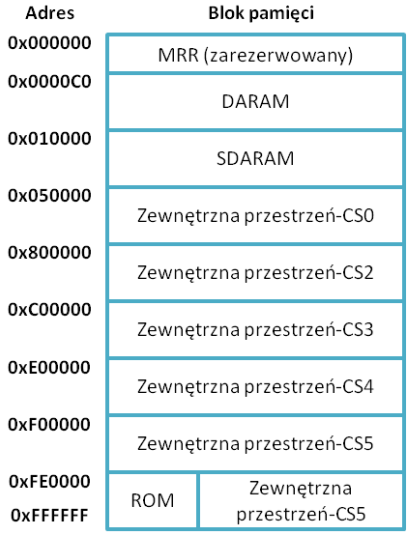
Rodzina C5000 niedrogich mikroprocesorów firmy Texas Instruments, dedykowana do zastosowań w przemyśle, charakteryzuje się niskim poborem mocy, stosunkowo dużą pamięcią wewnętrzną oraz dużym zestawem interfejsów komunikacyjnych. Ze względu także na dużą szybkość przetwarzania danych (taktowanie zegarem do 300 MHz) moduły elektroniczne z tymi procesorami są odpowiednie do zastosowań w systemach biometrycznych opartych m.in. na przetwarzaniu głosu. Przykładowym modułem jest TMS320C5515 eZDSP USB Stick [10] wykorzystywany przez autorów.

Mikroprocesor TMS320C5515 jest układem o architekturze 16-bitowej, przystosowanym do obliczeń stałoprzecinkowych (*fixed-point*). Taktowany jest zegarem o maksymalnej szybkości 120 MHz, co w przypadku instrukcji wykonywanych w czasie jednego i dwóch cykli zegarowych daje minimalny czas wykonania 8.33 ns. Układ jest wyposażony w dwie jednostki arytmetyczno-logiczne (ALU) oraz podwójną jednostkę mnożącą, pozwalające na maksymalnie 240 milionów operacji mnożenia i akumulacji w czasie 1 sekundy (MMAC). Duży wpływ na szybkość obliczeń ma akcelerator wyznaczania szybkiej transformaty Fouriera (FFT) o rozdzielczości od 2 do 1024 punktów. Jest to znaczące usprawnienie w opracowanym systemie przetwarzania mowy, w którym wykorzystano przekształcenie do dziedziny częstotliwości. Mikroprocesor posiada ponadto wbudowany interfejs obsługi wyświetlacza LCD oraz wejście kart pamięci SD podłączanych przez interfejs obsługi pamięci zewnętrznej (EMIF). Umożliwia szeregową transmisję danych według standardów I2C oraz SPI. Wykorzystany może być w bezprzewodowych urządzeniach audio, systemach usuwania pogłosu, w urządzeniach medycznych oraz w przemyśle. Na rysunku 1 przedstawiono schemat blokowy tego mikroprocesora.



Rys. 1. Schemat blokowy mikroprocesora TMS320C5515 [1]

Omawiany mikroprocesor jest wyposażony w 320 kB pamięci RAM, 128 kB pamięci ROM oraz 8/16-bitowy kontroler EMIF oraz DMA (16 kanałów). Wewnętrzna pamięć pozwala na akwizycję maksymalnie 20 sekund mowy przy założeniu, iż cała pamięć RAM będzie wykorzystana na gromadzenie danych o rozdzielczości 16 bitów z szybkością 8000 próbek na sekundę. Mapę pamięci prezentuje rys. 2. W przypadku modułu TMS320C5515 eZDSP USB Stick dostępna pamięć nieulotna FLASH ma wielkość 4 MB.



Rys. 2. Mapa pamięci mikroprocesora TMS320C5515 [10]

Mikroprocesor posiada także 4-kanałowy przetwornik analogowo-cyfrowy sukcesywnej aproksymacji (SAR ADC) o rozdzielczości 10 bitów, pracujący z szybkością 62,5 kS/s (32 cykle zegara taktowanego z szybkością 2 MHz na konwersję pojedynczej wartości napięcia wejściowego). Rozdzielczość oferowanego przetwarzania jest jednak zbyt mała do zastosowań audio, stąd moduł TMS320C5515 eZDSP USB Stick wyposażono

w zintegrowany koder dźwięku TLV320AIC3204, komunikujący się z mikroprocesorem z wykorzystaniem interfejsu szeregowego I2S (*Inter−IC−Sound*). Układ pracuje z szybkością od 8 do 192 kS/s, przesyłając dane mono lub stereo o rozdzielczości 16 bitów.

4. KONWERSJA OPROGRAMOWANIA ZE ŚRODOWISKA MATLAB DO ŚRODOWISKA CCS

Firma MathWorks będąca twórcą środowiska programistycznego Matlab, w którym zrealizowano i przetestowano opracowane algorytmy, umożliwia ich bezpośrednią konwersję do środowiska Code Composer Studio. W ten sposób otrzymuje się program uruchamiany na wybranym modelu mikroprocesora (z wielu dostępnych rodzin procesorów sygnałowych) firmy Texas Instruments [11]. Służy do tego aplikacja Matlab Coder. pozwalająca na generowanie kodu C i C++ zarówno na podstawie kodu w języku Matlab jak i algorytmów przygotowanych w środowisku Matlab / Simulink. Wygenerowany kod obsługuje większość funkcji języka Matlab takich jak funkcje i operacje na macierzach. Rozszerzenie to daje również możliwość zmiany struktury kodu tak, aby przyspieszyć krytyczne i czasochłonne obliczenia. Do kluczowych cech aplikacji Matlab Coder można zaliczyć:

- generację kodu C i C++ kompatybilną z ANSI/ISO
- generację kodu dla stało- i zmienno-przecinkowych obliczeń
- możliwość tworzenia bibliotek
- narzędzia do zarządzania strukturami i właściwościami danych
- statyczną lub dynamiczną alokację pamięci dla zmiennych
- wsparcie dla podstawowych funkcji języka Matlab (tj. operacji na macierzach, podprogramów, instrukcji sterujących, struktur, klas, liczb zespolonych i zmiennych globalnych), a także większości funkcji i obiektów wchodzących w skład następujących bibliotek: Communications System Toolbox, DSP System Toolbox, a także Computer Vision System Toolbox
- możliwość generowania kodu C z modeli Matlab / Simulink, które zawierają kod języka Matlab
- możliwość sprawdzenia poprawności wygenerowanego kodu jeszcze na poziomie języka Matlab
- możliwość wykorzystania wygenerowanego kodu do samodzielnie działających aplikacji (*standalone execution*), integracji z innym oprogramowaniem, przyspieszenia działania algorytmu poprzez wykorzystanie wykonywalnych funkcji języka C/C++ (MEX functions) oraz implementacji na procesorach sygnałowych.

Transformacja kodu z języka Matlab do C obejmuje określenie wymagań implementacyjnych. Matlab Coder jest narzędziem usprawniającym ten etap poprzez wyszukiwanie błędów, sprawdzanie składni oraz kompatybilności kodu z wybranym urządzeniem.

Jedną z części aplikacji Matlab Coder jest Embedded Coder, który umożliwia generację kodu wprost na procesory wbudowane, zintegrowane płytki służące do szybkiego prototypowania i mikropocesory produkowane masowo. Jego główną zaletą jest łatwa konfiguracja i dostosowanie dodatkowej optymalizacji wykorzystywanych funkcji, plików i danych. Kroki konwersji kodu Matlab do reprezentacji stałoprzecinkowej w C są następujące:

1. implementacja algorytmu w języku Matlab
2. przygotowanie testowej funkcji wywołującej zaimplementowany algorytm, w celu sprawdzenia zachowania algorytmu dla różnych danych wejściowych (stało- i zmiennoprzecinkowych) – test ten powinien zawierać jak najszerszy zakres wartości,

wygodne jest również wykorzystanie funkcji konwertującej dane ze zmienno-przecinkowych na stałoprzecinkowe
3. zbudowanie funkcji MEX dla algorytmu opracowanego w środowisku Matlab
4. uruchomienie programu dla minimalnych / maksymalnych wartości danych
5. sprawdzenie wygenerowanego przez Matlab Coder raportu
6. separacja kodu danych od kodu zmiennych
7. powtórne budowanie funkcji MEX oraz sprawdzenie poprawności jej działania tak długo, aż wszystkie błędy zostaną wyeliminowane a oprogramowanie zostanie zoptymalizowane
8. generacja ostatecznego kodu C.

Podczas prac eksperymentalnych przetestowano generację kodu C algorytmu wykorzystywanego do rozpoznawania mówcy w środowisku Matlab. Okazało się, że wadą takiego rozwiązania (korzystania z aplikacji Matlab Coder) jest generacja wielu plików w języku C. Przykładowo dla jednej z podstawowych funkcji *lmultigauss* (służącej do obliczania logarytmicznego prawdopodobieństwa z wielu mieszanin gaussowskich) otrzymano 41 plików o łącznej objętości 98 kB po konwersji do języka C. Kod tej funkcji w języku Matlab zajmował jedynie 2 pliki oraz 4 kB, przy czym należy pamiętać, że ten kod wykorzystuje kilka funkcji wbudowanych.

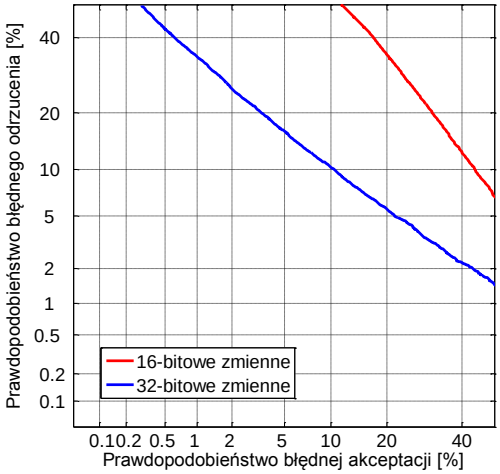
Dodatkową niedogodnością może okazać się to, iż pewne funkcje (takie jak *dot* czy *log*) nie są przystosowane w języku Matlab do operacji na zmiennych stałoprzecinkowych. W związku z tym adaptacja algorytmu do języka C wymaga modyfikacji tych funkcji w taki sposób, aby obsługiwały zmienne stałoprzecinkowe lub skorzystania z dodatkowych bibliotek.

5. WPŁYW REPREZENTACJI STAŁOPRZECINKOWEJ NA SKUTECZNOŚĆ IDENTYFIKACJI

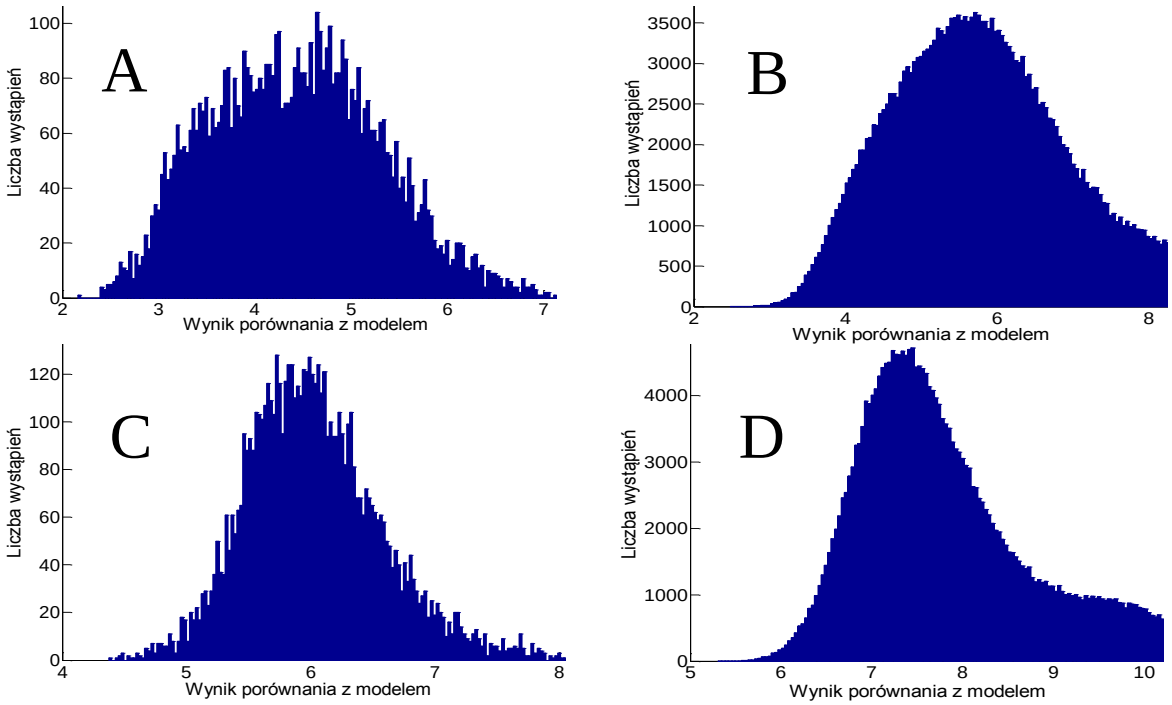
Do badań skuteczności identyfikacji wykorzystano autorską bazę mówców zawierającą nagrania 40 osób [9]. Każda z nich wypowiadała 6 krótkich zwrotów w języku polskim, powtarzając je dziesięciokrotnie w każdej z 3 sesji, między którymi interwały czasowe wynosiły od 2 do 6 tygodni. Daje to łączną liczbę 7200 nagrań próbkowanych z szybkością 22,05 kS/s w rozdzielczości 16 bitów. Do potrzeb eksperymentu, nagrania zostały zdecymowane do 8 kS/s. Badania wpływu reprezentacji stałoprzecinkowej odbywają się w dwóch etapach. W pierwszym, badana jest skuteczność rozpoznawania z wykorzystaniem notacji liczb 32-bitowych dla algorytmów kwantyzacji wektorowej (VQ) oraz mieszanin gaussowskich (GMM). Opis obu badanych algorytmów został przedstawiony w pracy [12]. W drugim etapie zmieniono rozdzielczość danych do 16 bitów.

Do pierwszego z eksperymentów wykorzystano algorytm VQ zaimplementowany w środowisku Matlab. Model każdego z mówców dla każdego ze słów był tworzony z trzech losowych nagrań, przy czym pozostałych 27 wykorzystano w fazie testu. Jako mierniki jakości rozpoznawania użyto zależności FAR/FRR (*false acceptance rate / false rejection rate*) oraz parametr EER (*equal terror rate*).

Wyniki eksperymentu przedstawia rys. 3. Algorytm bazujący na danych zmiennoprzecinkowych o rozdzielczości 32 bitów wykazuje się skutecznością mierzoną w oparciu o parametr EER na poziomie 10 %. Zmieniając reprezentację danych na stałoprzecinkową (16 bitów), zauważalny jest wzrost wartości EER nawet do 26 %. Rozkład wyników poprawnych i błędnych jest pokazany na rys. 4. Zmiana reprezentacji powoduje przesunięcie średniej wartości wyniku porównania oraz zwiększenie wariancji.



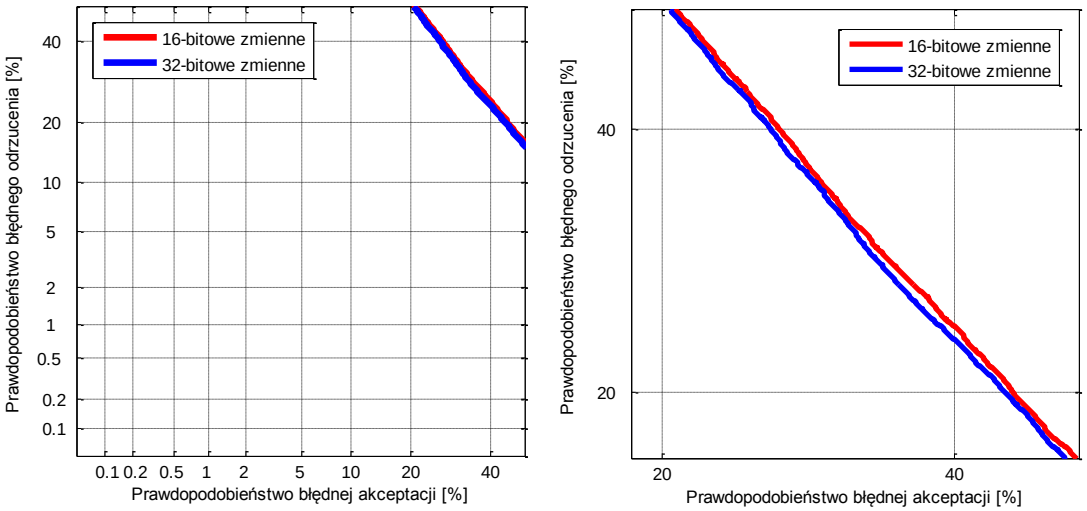
Rys. 3. Wykres FAR/FRR dla identyfikacji z wykorzystaniem algorytmu kwantyzacji wektorowej



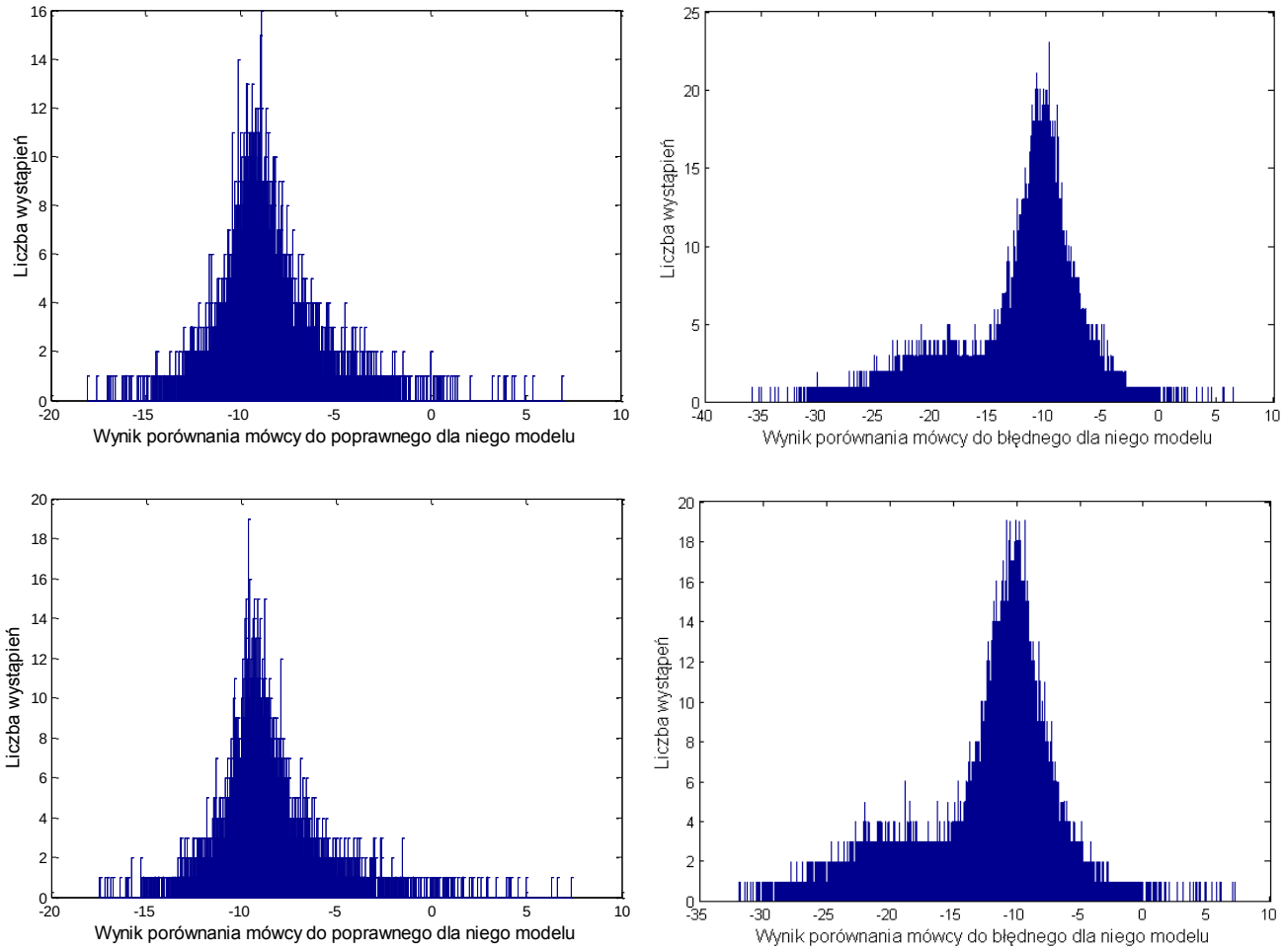
Rys. 4. Porównanie wyników (odległości Euklidesowych badanej próbki od modelu) dla reprezentacji 16-bitowych (poprawne – A, błędne - B) oraz 32-bitowych (poprawne – C, błędne – D) dla algorytmu VQ

Eksperyment wykorzystujący algorytm GMM zrealizowano w środowisku Matlab analogicznie jak opisany wyżej eksperyment z algorytmem VQ. W tym przypadku model mówcy dla każdego z 6-ciu wyrażeń również został utworzony z trzech losowych nagrań (27 pozostałych wykorzystano do testu). Otrzymane wyniki (wykres FAR/FRR) zostały pokazane na rys. 5. Parametr EER dla 32-bitowych danych zmiennoprzecinkowych wynosi 31 %. Redukcja rozdzielczości zmiennych do stałoprzecinkowych danych 16-bitowych ma nieznaczny wpływ na tę wartość. Parametr EER jest wówczas równy 32 %. Wynika to przede wszystkim z dużej gęstości danych w niewielkim przedziale liczbowym w przypadku danych 16 i 32-bitowych oraz z wąskiego zakresu wartości wyniku porównania testowych próbek mówcy z modelem. Rys. 6 przedstawia rozkład wyników porównań cech każdego mówcy do poprawnych i błędnych dla nich modeli. Na osi poziomej znajduje się wartość logarytmicznego prawdopodobieństwa pomiędzy modelem a testem obliczana na podstawie

wielu mieszanin gaussowskich. Zmiana reprezentacji powoduje nieznaczne przesunięcie skrajnych wartości wyników porównań oraz wariancji.



Rys. 5. Wykres FAR/FRR dla identyfikacji z wykorzystaniem algorytmu mieszanin gaussowskich (po lewej) oraz jego powiększenie (po prawej).



Rys. 6. Rozkład wyników porównań dla reprezentacji 16-bitowej (górne histogramy) oraz 32-bitowej (dolne histogramy) dla algorytmu GMM

PODSUMOWAMIE

Producenci procesorów sygnałowych starają się ułatwić proces implementacji algorytmów przetwarzania sygnałów mowy. Wbudowane elementy sprzętowe, a przede wszystkim przykładowe oprogramowanie oraz biblioteki, skracają proces projektowania ostatecznego systemu wbudowanego. Należy jednak pamiętać, że w pełni automatyczny proces konwersji oprogramowania ze środowiska Matlab do środowiska Code Composer Studio wymaga licznych ingerencji programistycznych i manualnej optymalizacji.

Zmniejszenie rozdzielczości zmiennych wykorzystywanych w algorytmie GMM nie ma znaczącego wpływu na jakość rozpoznawania mówcy. Można zatem wnioskować, iż przeniesienie algorytmu ze środowiska Matlab na stałoprzecinkowy 16-bitowy procesor nie wpływa znacząco na jakość działania systemu rozpoznawania mówcy.

BIBLIOGRAFIA

[1] Texas Instruments, TMS320C5515 Fixed-Point Digital Signal Processor, SPRS645E VIII 2010, REV I 2012, http://www.ti.com/lit/ds/symlink/tms320c5515.pdf .

[2] FFT Implementation on the TMS320VC5505, TMS320C5505, and TMS320C5515 DSPs (Rev. B) 09 Jan 2013.

[3] F. Bimbot, J. Bonastre, C. Fredouille, G. Gravier, I. Magrin-Chagnolleau, S. Meignier, T. Merlin, J. Ortega-Garcia, D. Petrovska-Delacretaz, and D. Reynolds, "A tutorial on text-independent speaker verification," in EURASIP Journal on Applied Signal Processing, 2004, pp. 430–451.

[4] Sadaoki Furui "50 years of progress in speech and speaker recognition" Proc. SPECOM 2005, Patras, Greece, pp.1-9 (2005-10).

[5] Jhing-Fa Wang; Jr-Shiang Peng; Jia-Ching Wang; Po-Chuan Lin; Ta-Wen Kuan, "Hardware/software co-design for fast-trainable speaker identification system based on SMO," Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on , vol., no., pp.1621,1625, 9-12 Oct. 2011.

[6] Y. S. Moon, C. C. Leung, K. H. Pun, Fixed-point GMM-based Speaker Verification over Mobile Embedded System, Proceeding WBMA '03 Proceedings of the 2003 ACM SIGMM workshop on Biometrics methods an, pp. 53-5

[7] Zhenling Zhang; Yangli Jia; Guang Xie, "Design and implementation of speaker recognition system," Software Engineering and Service Science (ICSESS), 2011 IEEE 2nd International Conference on , vol., no., pp.559,562, 15-17 July 2011.

[8] Maximiliano Lizondo, Pablo Daniel Agüero, Alejandro Jose Uriz, Juan Carlos Tulli and Esteban Lucio Gonzalez, Embedded speaker verification in low cost microcontroller Congreso Argentino de Sistemas Embebidos 2012. Buenos Aires, Argentina. 15-17 Agosto, 2012.pp. 128-133.

[9] Marciniak, T., Weychan, R., Drgas, Sz., Dąbrowski, A., Krzykowska, A., „Speaker recognition based on short polish sequences", Proc. of Signal Processing SPA'2010, Poland Section, Chapter Circuits and Systems IEEE, pp. 95-98, Poznań, Poland, September, 23-25th 2010.

[10] Spectrum Digital, TMS320C5515 eZDSP USB Stick Technical Reference, 512845-0001 Rev A II 2010
http://support.spectrumdigital.com/boards/usbstk5515/reva/files/usbstk5515_TechRef_RevA.pdf

[11] MathWorks, „Matlab Coder", http://www.mathworks.com/products/datasheets/pdf/matlab-coder.pdf

[12] T. Marciniak, A. Krzykowska, R. Weychan, „Speaker recognition based on telephone quality short Polish sequences with removed silence", Przegląd Elektrotechniczny, vol. 88, pp. 42-46, 2012.

Praca finansowana ze środków projektu INDECT.

Implementation of biometric processing of speech signal processing with the use of digital signal processor

Keywords: biometrics, speech processing, signal processor, Gaussian mixture models, vector quantization

ABSTRACT

This paper presents an analysis of issues related to the implementation of biometric speech signal processing based on fixed-point digital signal processor. During the experimental research, Matlab computing environment and development environment for digital signal processors Code Composer Studio (CCS) were used. Speech signal processing was carried out using a hardware module with TMS320C5515 processor. The aim of the system was speaker identification. The paper examines the limitations associated with the operation of the algorithm in an embedded system, demonstrates the technique of software converting from Matlab environment to CCS and shows the impact of fixed-point representation on the identification effectiveness.