Radosław Weychan, Tomasz Marciniak,
Adam Dąbrowski
Politechnika Poznańska,
Katedra Sterowania i Inżynierii Systemów
Pracownia Przetwarzania Sygnałów i Układów Elektronicznych
ul. Piotrowo 3a, 60-965 Poznań,
Radoslaw.Weychan@put.poznan.pl

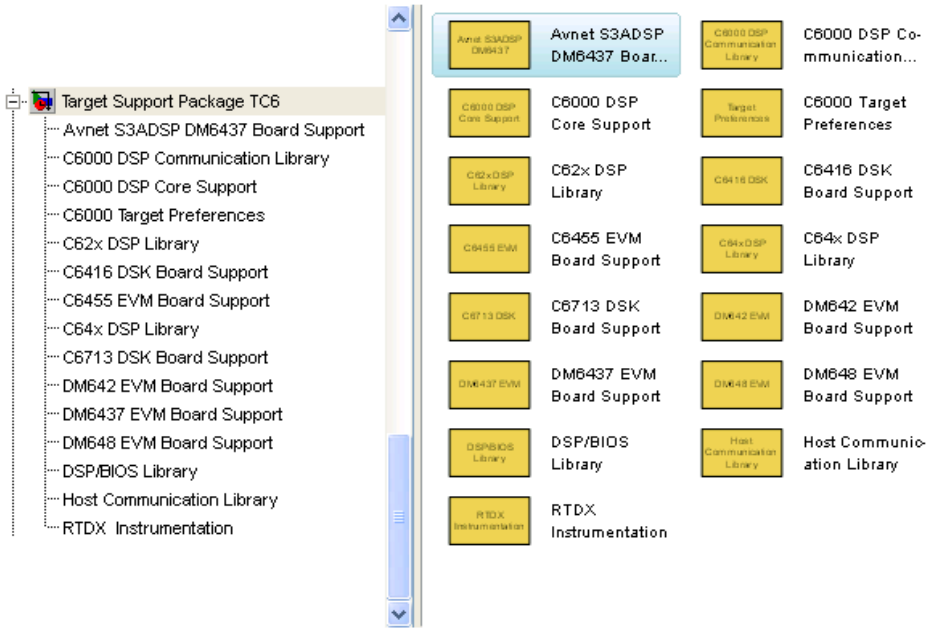
PRZETWARZANIE SYGNAŁÓW AUDIO W CZASIE RZECZYWISTYM Z ZASTOSOWANIEM TARGET SUPPORT PACKAGE TC6

Streszczenie – Artykuł przedstawia techniki przetwarzania sygnałów pasma akustycznego z wykorzystaniem modułu z procesorem sygnałowym, środowiska MATLAB/Simulink oraz biblioteki Target Support Package TC6. Pokazane modele, przygotowane dla procesora DM6437 w środowisku Matlab/Simulink, ilustrują proces implementacji algorytmów, korzystania z bibliotek oraz konfigurację bloków składowych modelu. Pokazano ułatwienia i ograniczenia przedstawionej techniki programowania.

1 Podstawowe cechy biblioteki Target Support Package TC6

Rozwój i coraz większe możliwości procesorów sygnałowych m.in. w przetwarzaniu sygnałów multimedialnych wymuszają opracowywanie nowych narzędzi ułatwiających programowanie systemów cyfrowego przetwarzania sygnałów (CPS). Nowoczesne środowiska programistyczne ułatwiają konfigurację sprzętu oraz dostęp do funkcji poprzez np. modułowość budowy. Jednym z takich środowisk jest Matlab/Simulink [1], który w połączeniu z biblioteką Target Support Package TC6 [3] pozwala na komunikację z modułem procesora sygnałowego z poziomu swojego interfejsu. Zawartość biblioteki TC6 ilustruje rys. 1.

W artykule szczególną uwagę zwrócono na moduły odpowiedzialne za akwizycję sygnałów audio (*DM6437 EVM Board Support*), konfigurację procesora sygnałowego (C6000 *Target Preferences* i *C6000 DSP Core Support*) oraz przetwarzanie odebranych z zewnątrz sygnałów (C64x *DSP Library*).



Rys. 1. Składniki biblioteki Target Support Package TC6

3.3 Akwizycja i przetwarzanie sygnałów audio z wykorzystaniem TC6

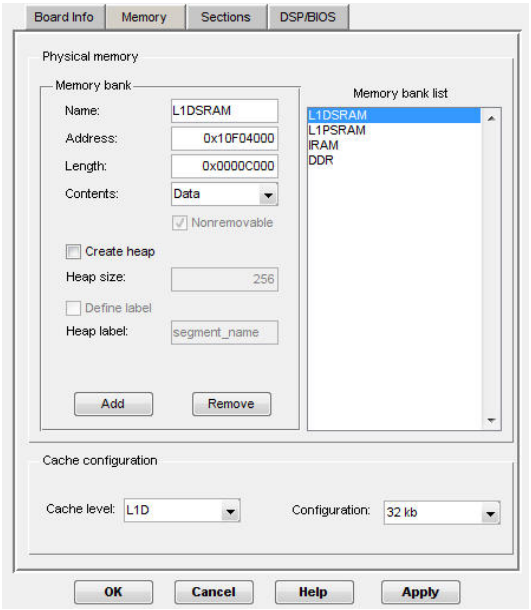
Target Support Package TC6 zawiera najważniejsze bloki dedykowane dla przetwarzania sygnałów audio, w tym mowy. Akwizycję sygnału umożliwiają moduły ADC / DAC (*DM6437 EVM Board Support*), które w przypadku procesora DM6437 EVM [2] pozwalają na konfigurację zintegrowanego kodeka AIC33 [3, 11]. Zapewniają dostęp do programowania szybkości próbkowania (8 kHz – 96 kHz) oraz do wewnętrznego bufora zawierającego ramki sygnału cyfrowego. W przypadku sygnału mowy ramki są typowo odpowiednikiem 10-20 ms sygnału ciągłego [9]. Rozdzielczość przetworników wynosi 16 bitów, stąd dane na wyjściu bloku ADC są typu INT16 [3]. Bloki ADC oraz DAC działają synchronicznie co oznacza, iż muszą mieć zadeklarowaną taką samą wartość szybkości próbkowania.

Realizację algorytmów cyfrowego przetwarzania sygnałów ułatwia biblioteka *C64x DSP Library*. Zawiera ona dedykowane i zoptymalizowane dla rodziny procesorów C64x bloki przetwarzania sygnałów. Wykonują one podstawowe operacje na wektorach i macierzach (m.in. autokorelację, zmianę typu danych, mnożenie, segmentację) oraz operacje wymagające większych mocy obliczeniowych, jak szybka transformacja Fouriera FFT czy filtracja za pomocą filtrów typu FIR lub IIR. Ze względu

na budowę procesora sygnałowego uwzględniają one ograniczenia związane z typem przetwarzanych danych. Powoduje to w niektórych przypadkach konieczność zmian reprezentacji danych lub wykorzystania bloków z biblioteki *Signal processing blockset* [4].

3.4 Konfiguracja procesorów rodziny C6000

Specyfika realizacji algorytmów z zastosowaniem procesorów sygnałowych wymaga konfiguracji parametrów związanych z alokacją danych w pamięci, szybkości pracy zegara, rozmiarem pamięci podręcznej typu cache oraz systemem BIOS. W przypadku procesorów z rodziny C6000 konfigurację taką umożliwia podbiblioteka C6000 Target Preferences. Umożliwia ona dostęp do ustawień mapy pamięci (zakładka *Memory*) oraz alokacji danych (zakładka *Sections* i *DSP/BIOS*). Przykładowe okno konfiguracyjne dla modułu DM6437 EVM prezentuje rys. 2.

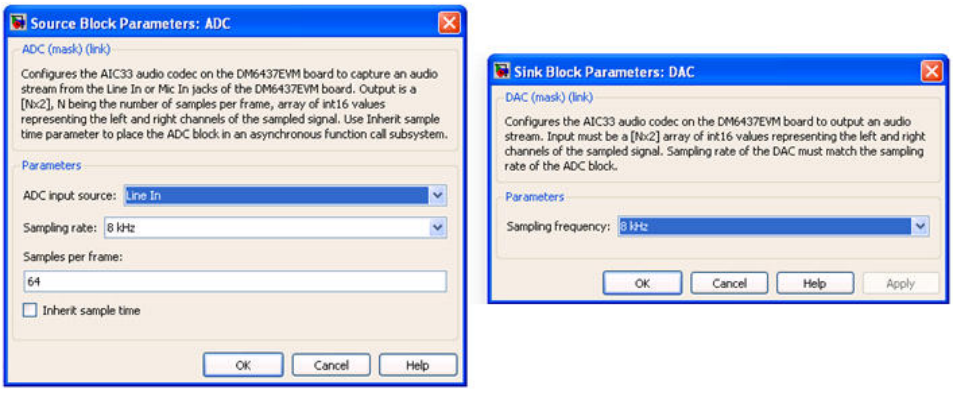


Rys. 2. Okno konfiguracji procesora sygnałowego

Alokację zmiennych w pamięci umożliwia blok *Memory Allocate* z podbiblioteki C6000 DSP Core Support. Dostępne w niej są także komponenty realizujące obsługę przerwań (*Hardware Interrupt*), tryb pracy modułów pamięci (*EDMA*), ustawienia priorytetów zadań (*Task* oraz *Idle Task*) oraz kopiowanie zawartości pamięci (*Memory Copy*).

Konfiguracja przetwarzania analogowo-cyfrowego w przypadku dźwięku, została także mocno ułatwiona. Standardowo należy wyzna-

czyć wartości rejestrów odpowiedzialnych za przekierowania sygnałów oraz dzielniki sygnału zegarowego w celu wyznaczenia według wzorów pożądanej szybkości próbkowania. W przypadku wykorzystania biblioteki TC6, procedura ta sprowadza się do ustawienia jedynie kilku wartości w osobnych oknach odpowiedzialnych za przetwarzania A/C oraz C/A. Konfigurowalnymi parametrami są: szybkość próbkowania (ze względu na synchroniczny sposób działania przetworników, wielkość ta musi być identyczna na wejściu oraz na wyjściu), źródło sygnału wejściowego oraz wielkość bufora odczytu. Okna te przedstawione są na rys. 3.



Rys. 3. Okna konfiguracji przetwarzania A/C oraz C/A

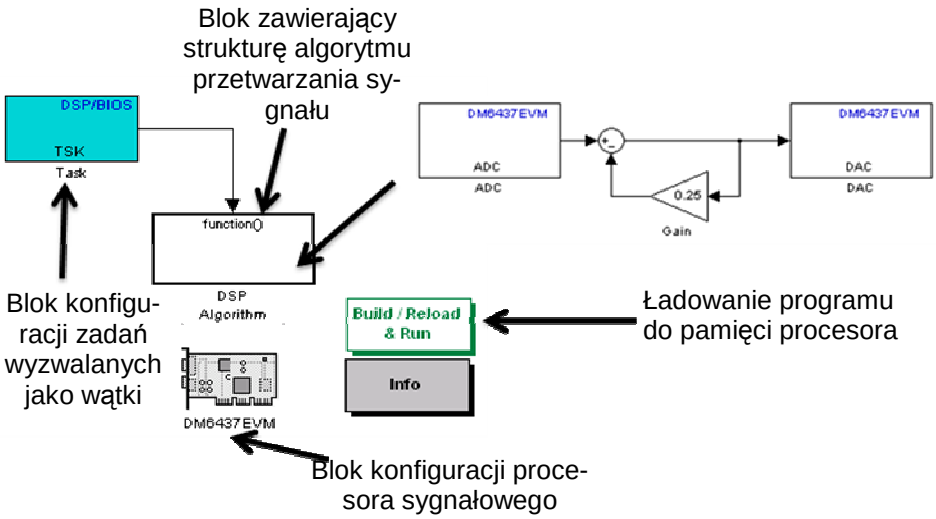
2 Modele blokowe w środowisku SIMULINK

Implementacja algorytmów cyfrowego przetwarzania sygnałów z zastosowaniem środowisk programistycznych dla procesora sygnałowego (np. Code Composer Studio [10] dla DM6437) jest żmudnym procesem wymagającym przygotowania plików z deklaracją mapy pamięci, wektorów przerwań, częstotliwości zegara taktującego i innych koniecznych do pracy parametrów. Dostęp do poszczególnych portów, w tym w szczególności do portu szeregowego wykorzystywanego przy akwizycji sygnału dźwiękowego, także musi zostać odpowiednio zaprogramowany z wykorzystaniem dzielników zegara głównego oraz rejestrów konfigurujących opcje przetwarzania. Typowo projekt składa się z kilkunastu lub kilkudziesięciu plików o niejednoznacznych typach powiązań między nimi.

Proces przygotowania oprogramowania z wykorzystaniem Target Support Package TC6 upraszcza i skraca czas implementacji algorytmów. Konfiguracja procesora sprowadza się do wstawienia na formatkę schematu bloku konfiguracyjnego dla danego typu procesora, którego

domyślne ustawienia sprawdzają się w większości zadań. Konfiguracja dostępu do poszczególnych portów także realizowana jest z wykorzystaniem bloków. Dla celów przetwarzania wielowątkowego pożądane jest określenie systemu nadzorującego pracę poszczególnych zadań, realizowanego przez blok BSP/BIOS widoczny na rys. 4.

Przejrzystość schematów blokowych jest dużo większa niż kodu języka C++, zwłaszcza w przypadku pętli oraz przekazywania parametrów między funkcjami. Tego typu sposób programowania zilustrowano przykładami w następnym rozdziale, natomiast ogólny wzór przygotowania schematu prezentowany jest na rys. 4.



Rys. 4. Wzorzec projektowy tworzenia schematów blokowych w środowisku Matlab/Simulink dla procesorów sygnałowych

3 Przykłady algorytmów przetwarzania sygnału mowy z zastosowaniem TC6

3.1. Wyznaczanie częstotliwości podstawowej tonu krtaniowego

Pierwszym przykładem ilustrującym sposób użycia biblioteki TC6 jest algorytm wyznaczania tonu podstawowego [7]. Działanie algorytmu polega na wyszukaniu indeksu z modułu wartości FFT o największej ampli-

tudzie. Częstotliwość tonu podstawowego obliczamy zgodnie ze wzorem (1).

$$F_0 = \frac{fs}{N} \times n \tag{1}$$

gdzie:

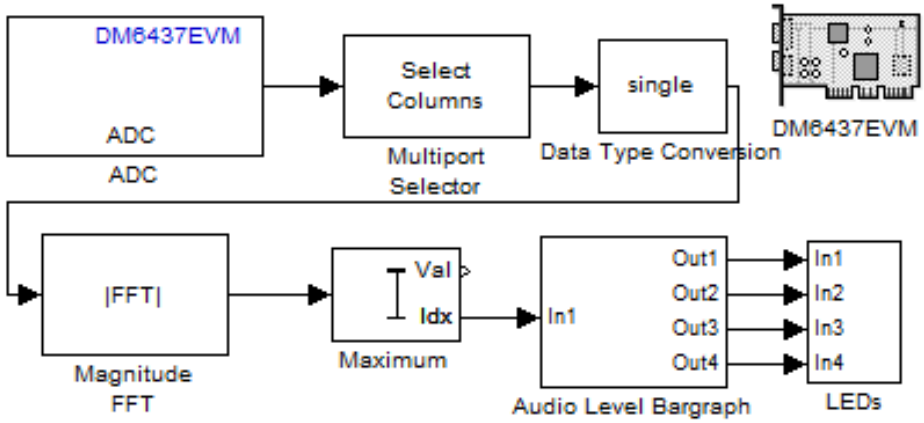
F_0 – częstotliwość podstawowa tonu krtaniowego

fs – szybkość próbkowania sygnału mowy

N – ilość punktów przekształcenia fft

n – numer prążka FFT o najwyższej amplitudzie.

Schemat blokowy algorytmu przedstawia rys. 5. Wykorzystano w nim blok akwizycji sygnału audio (ADC), połączony z *Multiport Select* w celu uzyskania sygnału monofonicznego. Zmiana typów danych blokiem *Data Type Conversion* z reprezentacji INT16 na single jest niezbędna ze względu na wykorzystanie funkcji obliczania modułu FFT. Indeks prążka o największej amplitudzie wybierany jest w bloku *Maximum*, a prezentacja wyniku odbywa się przykładowo na diodach LED.



Rys. 5. Schemat blokowy algorytmu wyznaczania tonu podstawowego

Przedziały przyporządkowania zakresów częstotliwości do wizualizowanej wartości przedstawia tabela 1. Rozdzielczość transformaty FFT wynosi 62,5 Hz i stąd wynikają granice przedziałów będące wielokrotnościami tej wartości.

Tabela. 1. Przyporządkowanie wartości częstotliwości tonu do przedziałów

Częstotliwość podstawowa tonu [Hz]	Przyporządkowany przedział [Hz]
200	0-312,5
400	312,5-562,5
600	562,5-750
800	750-4000

3.2 Wyznaczanie parametru SNR z wykorzystaniem interfejsu komunikacji RTDX

Kolejnym przykładem, który można zrealizować z zastosowaniem biblioteki TC6 jest wyznaczanie SNR (ang. *Signal to Noise Ratio*) w czasie rzeczywistym. Pierwotny model przygotowany dla procesora TMS320C6713 został zaadaptowany dla modułu DM6437 EVM. Stosunek sygnału do szumu wyznaczany jest z jako stosunek wartości oczekiwanej do odchylenia standardowego.

$$SNR = \frac{\mu}{\sigma} \tag{2}$$

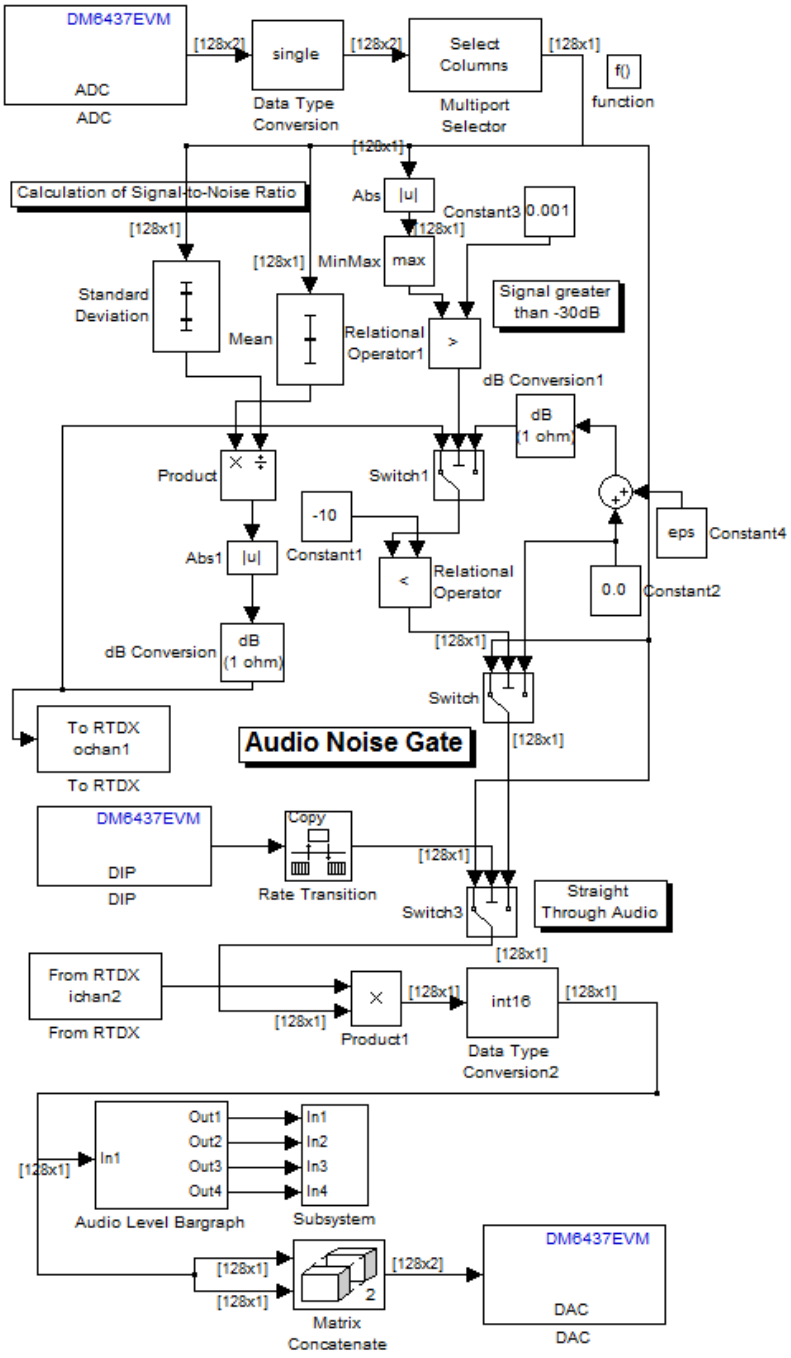
gdzie:
μ - wartość oczekiwana
σ - odchylenie standardowe.

Bufor wejściowy dla bloku ADC ustawiono na 128 próbek (ramki 16 ms), co gwarantuje poprawne działanie algorytmu dla sygnału mowy.

W celu wyznaczenia stosunku sygnału do szumu należy użyć funkcji z biblioteki Signal Processing. Wykorzystano bloki obliczające odchylenie standardowe, wartość oczekiwaną, konwersję na skalę decybelową, konwersję formatów oraz obliczenia arytmetyczne. Dodatkowo na wyjście jest wyprowadzany jedynie sygnał, którego aktualny poziom jest większy niż -30 dB. Schemat blokowy dla środowiska Simulink prezentuje rys. 6.

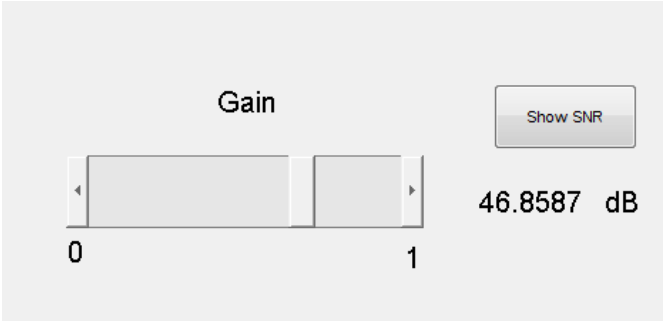
Odczyt wartości SNR odbywa się z poziomu osobnego programu GUI (ang. *Graphical user interface*) [5] (rys. 7), z wykorzystaniem polecenia *readmsg* (*nazwa_kanału, typ_zmiennej)* umieszczonego w kodzie programu. Nazwa kanału musi być zadeklarowana w modelu Simulink w bloku *To RTDX*, za pomocą którego transmitowane są dane [3,6]. Mimo iż kolejne wersje pakietu TC6 nie wspierają możliwości tego interfejsu,

dla wielu wykorzystywanych aktualnie procesorów sygnałowych jest to najszybszy i najprostszy sposób komunikacji ze środowiskiem zewnętrznym przez port USB.



Rys. 6. Schemat blokowy algorytmu obliczania SNR

Program GUI umożliwia także regulację głośności na wyjściu przesyłając w czasie rzeczywistym współczynnik wzmocnienia również za pomocą kanału RTDX (blok *From RTDX*). W tym przypadku należy również dokonać deklaracji nazwy kanału. Szybkość transmisji wynosi 20÷50 kbps.



Rys. 7. Graficzny interfejs sterowania współczynnikiem wzmocnienia oraz odczytu współczynnika SNR

Środowisko Matlab/Simulink wspomaga konwersję między typami zmiennych. Wykorzystane w przykładach bloki *Data Type Conversion* oraz *dB Conversion* umożliwiają szybką i łatwo konfigurowalną zmianę typów danych. Złożone operacje na macierzach umożliwia m. in. blok *Matrix concatenate*.

3.3 Sterowanie parametrami przetwarzania na przykładzie parametryzacji współczynników opóźnień

Na rys. 8. przedstawiono model realizujący funkcję pogłosu [9]. Blok Reverbation Algorithm realizujący tę funkcję opisany jest następującym równaniem różnicowym

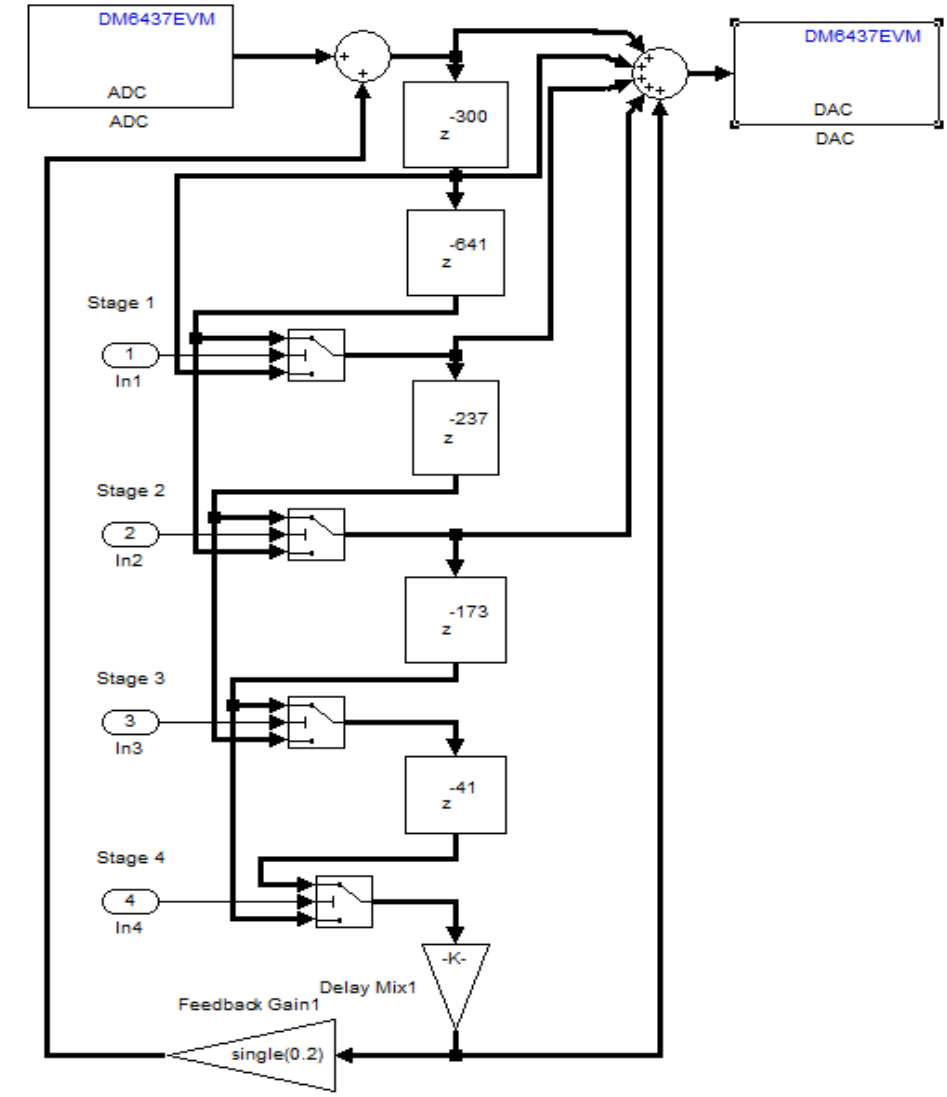
$$y(n) - 0{,}12\,y(n-1392) = x(n) + x(n-300) + \\ + x(n-941) + x(n-1178) + 0{,}6x(n-1392) \tag{3}$$

Realizacja równania w środowisku Matlab/Simulink umożliwia budowę schematu w formie bezpośredniej kanonicznej prezentowanej na rys. 8 jako zawartość bloku Reeverbation Algorithm. Dostęp do poszczególnych stopni opóźnień odbywa się z poziomu zintegrowanych z modułem DM6437 EVM przełączników (SW4, *switches* 0-3). Przełączniki te pozwalają na wprowadzenie lub usunięcie dodatkowego opóźnienia.

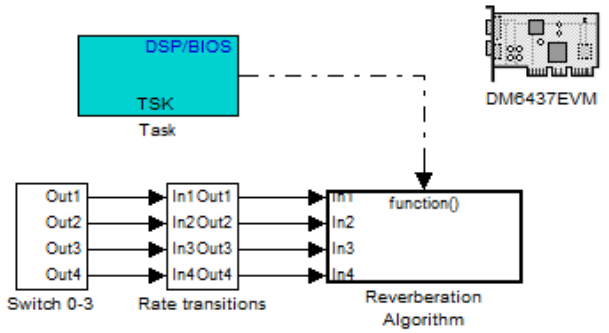
Wszystkie zadania zarządzane są przez system BIOS, widoczny na rys. 9.

Wykorzystanie schematu blokowego w przeciwieństwie do języka wysokiego/niskiego poziomu ułatwia orientację w algorytmie i przyspiesza modyfikacje oraz wprowadzanie dodatkowych elementów.

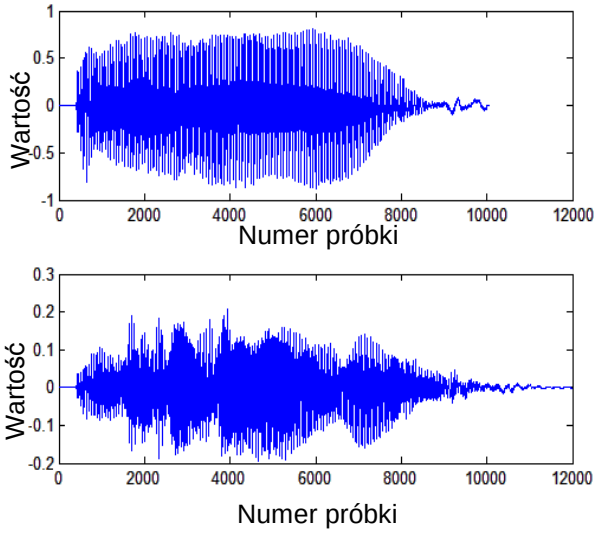
Na rys. 10 przedstawiono działanie algorytmu na przykładzie głoski ‘a’ oraz włączonych wszystkich opóźnieniach.



Rys. 8. Forma bezpośrednia kanoniczna równania różnicowego



Rys. 9. Schemat główny programu



Rys. 10. Głoska nieprzetworzona (wykres górny) oraz poddana operacji filtracji (wykres dolny)

3.4 Efektywność wykorzystania mocy obliczeniowej procesora sygnałowego

Procentowe obciążenie procesora wyznaczone dla każdego z prezentowanych przykładów prezentuje tabela 2.

Uzyskane wartości obciążenia, biorąc pod uwagę częstotliwość zegara taktującego równą 594 MHz, pozwalają wnioskować o nieoptymalności wygenerowanego przez automat kodu. Ciekawym przykładem jest wykorzystanie transformaty FFT, dla której obciążenie procesora sięga nawet 25 %.

Tabela. 2. Obciążenie procesora sygnałowego dla prezentowanych przykładów

Przykład	Minimalne obciążenie [%]	Maksymalne obciążenie [%]
Wykrywanie częstotliwości podstawowej tonu krtaniowego	4,47	25
Obliczanie SNR	9,74	9,85
Realizacja pogłosu	6,64	6,84

4 Podsumowanie

Środowisko Matlab/Simulink wraz z biblioteką Target Support Package TC6 umożliwia realizację algorytmów na procesorach sygnałowych oraz ich konfigurację w niewykorzystywany do tej pory sposób. Zalety oraz ograniczenia takich reguł projektowania systemów zostały przedstawione w zestawieniu poniżej.

Zalety:
1. Krótki czas przygotowania projektu
2. Prosta konfiguracja procesora oraz bloków akwizycji i przetwarzania sygnałów
3. Intuicyjny interfejs
4. Duża przejrzystość schematów
5. Modułowa generacja kodu w języku C++ z komentarzami

Ograniczenia:
1. Brak optymalizacji kodu oraz niska szybkość działania w przypadku złożonych modeli
2. Brak przystosowania elementów biblioteki Signal Processing do pracy ze wszystkimi modelami procesorów
3. Konieczność odgórnej deklaracji wielkości tablic
4. Problemy z realizacją sprzężeń zwrotnych

Środowisko Matlab/Simulink wraz z biblioteką Target Support Package TC6 jest narzędziem ułatwiającym implementację oraz testowanie algorytmów przetwarzania sygnału mowy. Dostępne w nim funkcje umożliwiają szybkie przygotowywanie własnych aplikacji oraz dostęp do

danych zarówno w celu ich odczytu jak i konfiguracji w czasie rzeczywistym. Należy jednak mieć na uwadze fakt, iż jest to przede wszystkim środowisko pomocnicze i wygenerowany kod języka wysokiego poziomu nie jest zoptymalizowany pod względem wykorzystania mocy obliczeniowej procesora sygnałowego.

Głównym zastosowaniem biblioteki TC6 jest możliwość szybkiej weryfikacji algorytmów pod względem poprawności obliczeniowej. Optymalizację kodu ułatwiają komentarze oraz modułowość wygenerowanych projektów, jednakże w wielu przypadkach jest to nieopłacalne ze względu na konieczność przebudowy całego oprogramowania.

Literatura

[1] MATLAB® 7 Getting Started Guide, Mathworks, 2009

[2] TMS320DM6437 DVDP Getting Started Guide, Texas Instruments, 2007

[3] Target Support Package™ 3.6 User's Guide, Mathworks, 2009.

[4] Signal Processing Blockset™ 6 User's Guide, Mathworks, 2009.

[5] Creating GUI's in Matlab, MeghanStephens, April 2,2007

[6] From MATLAB® and Simulink® to Real Time with TI DSPs, Texas Instruments Inc, 2007.

[7] Tadeusiewicz R., *Sygnał mowy,* Wydawnictwa Komunikacji i Łączności, Warszawa,1988.

[8] Lyons R., *Wprowadzenie do cyfrowego przetwarzania sygnałów,* Wydawnictwa Komunikacji i Łączności, Warszawa 2000.

[9] Zieliński T. P., *Cyfrowe przetwarzanie sygnałów: od teorii do zastosowań*, Wydawnictwa Komunikacji i Łączności, Wyd. 2 popr, Warszawa 2007.

[10] Code Composer Studio Development Tools v3.3Getting Started Guide, Texas Instruments Inc, 2006.

[11] TLV320AIC33/3106/34 Stereo Audio Converters, Texas Instruments Product Bulletin, 2007.

W pracy zaprezentowano wyniki osiągnięte w projektach PPBW, INDECT I BW.

REAL TIME AUDIO SIGNAL PROCESSING USING TARGET SUPPORT PACKAGE TC6 LIBRARY

Summary – This paper describes techniques of acoustic signal processing using a digital signal processor module, MATLAB/Simulink environment and Target Support Package TC6 library. The presented models, prepared for DM6437 signal processor, illustrate the use of libraries and the configuration of the model blocks. Facilities and limitations of the presented programming techniques are also shown.