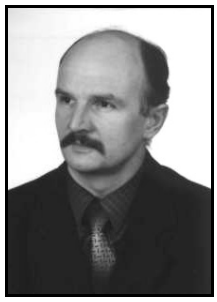
Zenon SYROKA
UNIWERSYTET WARMIŃSKO – MAZURSKI W OLSZTYNIE, WYDZIAŁ NAUK TECHNICZNYCH, KATEDRA ELEKTROTECHNIKI I ENERGETYKI
Oczapowskiego 11, 10-719 Olsztyn

Wykorzystanie szybkiej transformaty Walsha (Fast Walsh Transform) do budowy programowanych kluczy zabezpieczających

Dr inż. Zenon SYROKA



Jest adiunktem w Katedrze Elektrotechniki i Energetyki UWM w Olsztynie. Ukończył Wydział Elektroniki WAT w Warszawie i Wydział Matematyki i Informatyki UMK w Toruniu. Jego zainteresowania naukowe to: radiokomunikacja; systemy telekomunikacyjne; analiza, przetwarzanie sygnałów, obrazów i drgań; zastosowania matematyki i informatyki w naukach technicznych; sterowanie maszyn; układy elektroniczne analogowe i cyfrowe; programowane układy elektroniczne; programowanie C, PASCAL, MATLAB, VHDL.

e-mail: syrokaz@onet.eu

Streszczenie

Praca napisana jest na podstawie zgłoszenia patentowego [1] "Sposób analizy i rozpoznawania mowy oraz układ do analizy i rozpoznawania mowy". Zbudowano układ do analizy i rozpoznawania mowy przy pomocy szybkiej transformaty Walsha. Założono możliwie jak najmniejszy koszt konstrukcji, dlatego wykorzystano procesor AT89S51. W niniejszym artykule przedstawiono opis układu do analizy i rozpoznawania mowy oraz implementację algorytmu FWT na procesor AT89S51. Urządzenia wykonawcze – blokujące dostęp, są różne i zależą do rodzaju maszyny i w związku z tym nie będą w pracy opisywane.

Słowa kluczowe: klucze sprzętowe, szybka transformata Walsha, programowanie procesorów, rozpoznawanie sygnałów mowy.

Use of Fast Walsh Transform for building programmable security keys

Abstract

This work is based on patent application "The way to analyze and recognize the speech and module to analyze and recognize the speech". The module to analyze and recognize the speech was built using the Fast Walsh Transform. The idea was to reduce construction cost. That is why a processor AT89S51 was used. In Section 2 of this paper there is contained description of the module for speech analysis and recognition whose block diagram is shown in Fig. 1. Fig. 2 presents implementation of the FWT algorithm on an 8bit processor AT89S51. Section 3 focuses on the algorithm of the FWT procedure. The Fast Walsh Transform consists of three subprocedures (MINUS, PLUS - Fig. 4, ZAPIS - Fig. 5), which are generated in a specific order. The executive devices – blocking access are different and depend on specification of the machine specification so they are not described in this work.

Keywords: programmable security keys, Fast Walsh Transform, speech recognition.

1. Wstęp

W klasycznej analizie sygnałów mowy najczęściej wykorzystywane jest przekształcenie STFT (Short Time Fourier Transform). Widmo otrzymanego sygnału znajduje się w dziedzinie częstotliwości. Należy również zaprojektować odpowiednie okna czasowe oraz użyć algorytmu szybkiej transformaty Fouriera (FFT). Funkcjami bazowymi w analizie fourierowskiej są funkcje trygonometryczne, co daje dużo mnożeń przez funkcję ekspotencjalną. Powoduje to wydłużenie czasu analizy.

Praktyczna analiza sygnałów w czasie rzeczywistym wymaga krótkiego czasu wykonywania obliczeń. Do takiej analizy bardzo dobrze nadają się funkcje Walsha, które tworzą bazę ortogonalną [5, 6, 7, 9]. Posiadają one wartość plus jeden albo minus jeden. Operacja mnożenia przez odpowiednią funkcję bazową polega na przemnożeniu wejściowych próbek sygnału przez jeden lub minus jeden, co znacznie przyśpiesza analizę. Przy pomocy transformat Walsha [5, 6, 7, 8, 9] dokonuje się analizy sekwencyjnościowej sygnałów mowy. Algorytmy szybkiej transformaty Walsha zaprezentowane są w literaturze [5, 7, 8, 9]. Widmo powstające w wyniku tych przekształceń jest reprezentacją sygnału w dziedzinie uogólnionej częstotliwości, zwanej sekwencyjnością [5, 7, 9]. Sekwencyjność jest podstawowym parametrem charakteryzującym funkcje Walsha. Jest ona ściśle związana z liczbą przejść funkcji Walsha przez zero.

Przy budowie układów rozpoznających mowę w oparciu o rozkłady względem funkcji bazowych Walsha wykorzystuje się współczynniki rozwinięcia w szereg lub jak w tym przypadku współczynniki szybkiej transformaty Walsha (FWT) zapisane w postaci wektora lub macierzy.

Budując układ [1] autor przeanalizował inne rozwiązania patentowe związane z przetwarzaniem sygnałów mowy.

I tak np. w patencie nr 160852 pt. „Sposób i układ do przesyłania sygnału mowy" po przekształceniu sygnału na postać cyfrową mierzy się długości interwałów czasowych między kolejnymi przejściami przez zero i określa się amplitudy w tych interwałach. Następnie zmierzone interwały i amplitudy przesyła się do odbiornika, gdzie ciągi par liczb interwałów i amplitud poddaje się resyntezie. Oblicza się dyskretne wartości chwilowe sygnału dla każdego interwału z uwzględnieniem wartości i znaku amplitudy.

Znany jest również z opisu patentowego nr 193825 pt „Sposób i urządzenie do kodowania mowy" sposób kodowania sygnałów mowy ze zmienną prędkością przesyłania bitów.

Rozwiązania te mają zastosowania głównie do transmisji sygnałów mowy. W obydwu przypadkach istnieje algorytm rekonstrukcji sygnału.

Autor również w [10] wykorzystał układ klasycznych wielomianów ortogonalnych (Czebyszewa I rodzaju, Laguerra, Hermita, Legendra) do budowy bezprzewodowego systemu diagnostycznego. W systemie tym istniała również potrzeba rekonstrukcji sygnału.

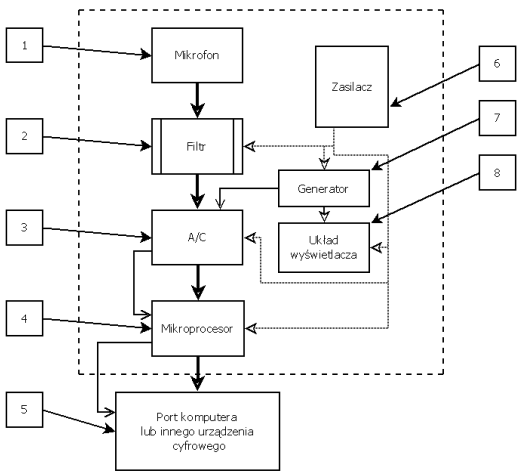
W proponowanym układzie do analizy i rozpoznawania mowy algorytm rekonstrukcji jest zbędny. W związku z tym analiza sygnału może zostać uproszczona do obliczenia współczynników FWT. Nie jest potrzebna dokładna znajomość częstotliwości formantów, charakterystyk fonemów czy zespołów fonemów. Wystarczy dokonać rozkładu sygnału mowy względem dowolnego układu funkcji tworzących układ ortogonalny. Układ ten powinien być dogodny do implementacji programowej z wykorzystaniem możliwie najprostszego mikroprocesora. W tym przypadku wykorzystano funkcje Walsha tworzących układ ortogonalny.

Budowa zabezpieczeń z użyciem hasła wypowiedzianego przez użytkownika polega na tym, iż sygnał wzorcowy (hasło) zostaje przetransformowane w dziedzinę sekwencyjności (FWT) i zapisane w pamięci systemu. Hasło wypowiedziane przez dowolnego innego użytkownika również zostaje przetransformowane w dziedzinę sekwencyjności i zapisane w pamięci. Następnie zostają porównane oba zapisy. W zależności od kryterium zgodności i dokładności obu przebiegów jest podjęta decyzja, co do autentyczności osoby posługującej się hasłem, a co za tym idzie, wysyłany jest sygnał blokady, ewentualnie uruchomienia urządzeń lub maszyn.

2. Opis układu do analizy i rozpoznawania mowy

Schemat blokowy układu do analizy sygnału mowy pokazano schematycznie na rys. 1. Sygnał wejściowy z mikrofonu jest filtrowany przez środkowoprzepustowy filtr aktywny 2. Następnie poddawany jest próbkowaniu w przetworniku A\C (3) sterowanym sygnałem generatora 7. Po kwantyzacji próbki zostają wysłane do mikrokontrolera 4 sterowanego przetwornikiem A/C (3)

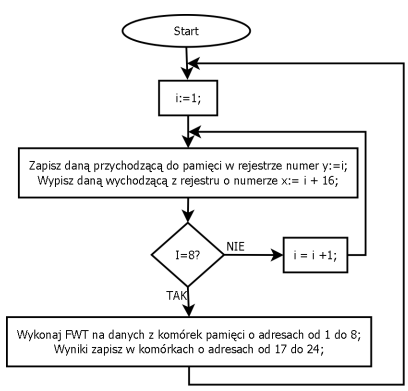
gdzie dokonywana jest transformacja w dziedzinę sekwencyjności (FWT). Po zakończeniu transformacji, dane wysyłane są do urządzeń zewnętrznych 5 w postaci cyfrowej. Urządzenia zewnętrzne 5 sterowane są sygnałem mikrokontrolera 4. Wszystko jest zasilane z zasilacza 6. Częstotliwość próbkowania jest mierzona częstotliwościomierzem i wyświetlana na wyświetlaczu segmentowym w obrębie układu wyświetlacza 8.



Rys. 1. Schemat blokowy układu do analizy i rozpoznawania mowy
Fig. 1. Block diagram of the module for speech.analysis and recognision

Układ realizujący szybką transformatę Walsha zbudowano wykorzystując typowy najprostszy i najtańszy układ ośmiobitowy AT89s51 [1].

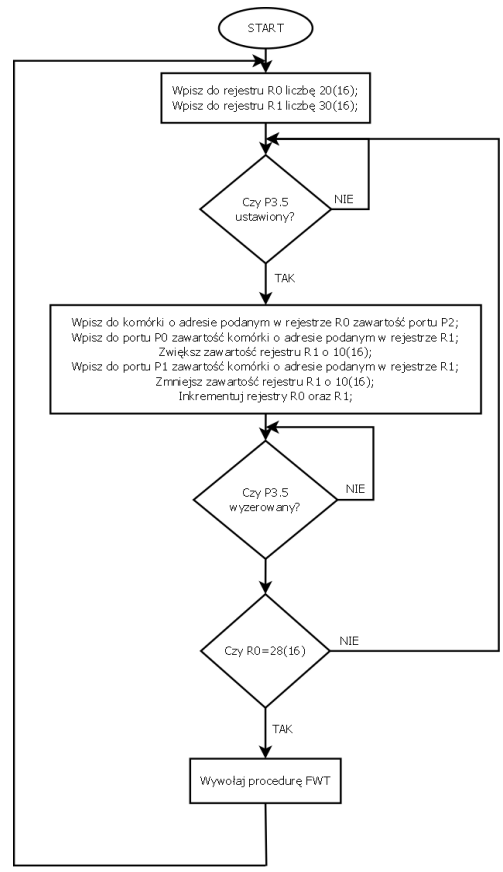
Program od początku działania będzie realizował pobieranie oraz wystawianie danych na przemian. Po zapisaniu zestawu 8 bajtów danych przejdzie do transformacji używając procedury FWT. Następnie wróci do początku i ponowi pobieranie zestawu 8 bajtów danych na przemian wystawiając dane. Do pobierania danych z przetwornika A/C wystarczy jeden port mikrokontrolera, natomiast dane wychodzące będą miały postać 12 bitową. Za pomocą liczby 12 bitowej możemy przedstawić 4096 stanów. Jest to wystarczająco dużo żeby zapisać wynik FWT. Schematyczne działanie programu pokazano na rys. 2.



Rys. 2. Algorytm programu. FWT dla mikroprocesora ośmiobitowego
Fig. 2. FWT algorithm for an 8 bit microprocessor

Główny program składa się z 3 pętli, kilku linijek kodu i procedury FWT, rys. 3. Od początku działania zapisuje on dane z portu do pamięci i jednocześnie wypisuje dane przetransformowane, zwracając uwagę na stan rejestru P3.5 ustawianego przetwornikiem. Po zapisaniu 8 bajtów wywołuje procedurę Szybkiej Transformaty Walsha.

Po włączeniu układu mikroprocesora AT89s51 program wpisuje do jego rejestrów R0 oraz R1 liczby w systemie szesnastkowym o wartościach dla R0 – $20_{(16)}$ a dla R1 – $30_{(16)}$. Rejestry te zostaną wykorzystane do pośredniego adresowania komórek pamięci, w których zapisywane i odczytywane będą dane. Po wykonaniu tych operacji realizowana jest pętla, która oczekuje na ustawienie stanu wysokiego wyprowadzenia P3.5. Po pojawieniu się stanu wysokiego na tej linii pochodzącego od ADC, mikrokontroler przechodzi do wykonywania kolejnych instrukcji. Mianowicie rozpoczyna się pobieranie oraz wystawianie kolejno 8 bajtów danych przychodzących oraz wychodzących. W tym celu wykorzystano pętlę powtarzającą się osiem razy. Pętla ma postać „Repeat …. Until ….". W pierwszej kolejności wykonywane są instrukcje a na końcu sprawdzany jest argument.



Rys. 3. Algorytm programu głównego.
Fig. 3. The algorithm of the main program

Przy pierwszym powtórzeniu pętli do komórki o adresie $20_{(16)}$– wskazywanym przez rejestr roboczy R0 zostaje zapisany bajt pochodzący od przetwornika A/C z rejestru portu P2. Do rejestru portu P0 zostaje wpisana zawartość rejestru $30_{(16)}$ – wskazywanego przez rejestr R1, a następnie rejestr ten zostaje zwiększony o $10_{(16)}$. Komórka o adresie wskazywanym przez R1 (40h) zostaje wystawiona na port P1. Jak już wcześniej wspomniano na port P0 są wystawiane mniej znaczące bity danych przetransformowanych, natomiast na port P1 bardziej znaczące bity. Po tych instrukcjach rejestr R1 jest zmniejszany o 10h, a następnie oba rejestry R1 oraz R0 zostają poddane inkrementacji. Na tym etapie program oczekuje na pojawienie się niskiego stanu na wyprowadzeniu P3.5. Zapobiega to wpisaniu do pamięci kilkukrotnie a nawet kilkunastokrotnie tej samej danej. Taka sytuacja może zaistnieć przy dużej częstotliwości taktowania mikroprocesora, gdy program powróci do początku działania i sprawdzi stan P3.5, który będzie wysoki.

Tak więc, przy pomocy pętli oczekujemy na stan niski rejestru portu P3.5. Następnie przechodzimy do sprawdzenia argumentu pętli. Argumentem jest rejestr R0 równy $28_{(16)}$ (R0=28h). W przypadku spełnienia tej równości program przechodzi do wywołania procedury FWT, każda inna możliwość (R0<28h) powoduje powrót do początku pętli, która zaczyna się od oczekiwania na stan wysoki rejestru P3.5. Tym razem dane zapisane zostaną do komórki o adresie 21h a wypisane zostaną z komórek o adresach 31h oraz 41h.

Kolejno program inkrementuje wskaźniki, czeka na stan niski P3.5 sprawdza argument R0=28h i wraca do początku, gdy ten nie

jest spełniony. Po sześciokrotnym wykonaniu się pętli rejestr R0 równy jest 27h. Dane zapisane zostają do komórki o adresie 27h, wypisane zostają dane z komórek 37h oraz 47h. Inkrementowany następnie rejestr R0 posiada wartość 28h. Zakończenie pętli sprawdza warunek i przechodzi do wywołania procedury FWT. Zapisane 8 bajtów danych w komórkach od 20h do 27h zostaje wykorzystane w procedurze FWT.

Wszystkie dane wychodzące z komórek od 30h do 37h oraz od 40h do 47h zostały przesłane do kolejnych urządzeń poprzez działanie pętli, nie mamy więc obaw o utratę tych danych i wyniki otrzymane w procedurze FWT zapisane zostają w tych rejestrach.

3. Algorytm procedury FWT

Procedura szybkiej transformaty Walsha (FWT) jest ściśle związana z macierzą tej transformaty [8, 9] i została ona zaimplementowana na mikroprocesorze AT89s51. Składa się ona z trzech podprocedur (MINUS, PLUS, ZAPIS), które zostają wywoływane w odpowiedniej kolejności. Na samym początku procedury FWT realizowane są operacje przygotowujące zsumowanie zmiennych z pierwszego wiersza macierzy, następuje wpisanie liczby 1020 do rejestru 16 bitowego składającego się z rejestrów R6 oraz R7. Rejestr R6 przechowuje mniej znaczący bajt, natomiast R7 bardziej znaczący. Oba rejestry służą przechowywaniu wyniku działań. Następnie do rejestrów R0 oraz R1 wpisane zostają adresy: do R0 – komórki zawierającej wartość pierwszej zmiennej (20h), do R1 – komórki, w której zostanie zapisany pierwszy wynik (młodszy bajt – 30h, starszy bajt generowany programowo przez dodanie 10h). Rejestr R0 przechowujący adres zmiennej przy działaniu na wierszu jest inkrementowany od wartości 20h do 27h. Są to wartości próbek od a_0 do a_7. Rejestr R1 przechowuje adres, pod którym zapisany zostanie mniej znaczący bajt wyniku działań na każdym wierszu. Adres do zapisu starszego bajtu uzyskujemy przez zwiększenie R1 o 10h. Tak wiec wartości R1 to liczby od 30h do 37h oraz od 40h do 47h. Działanie na wierszach odbywa się poprzez wywołanie w odpowiedniej kolejności procedur PLUS lub MINUS. Kolejność wywołania zależy od wymnożonej macierzy FWT.

Gdy w wierszu, na którym działamy mamy liczbę sumowaną, wywoływana jest procedura PLUS, w przypadku liczby odejmowanej wywoływana jest procedura MINUS. Wywołując zaczynamy od a_0 do a_7. Procedury te dodają/odejmują daną wskazywaną przez rejestr R0 do/od rejestrów R6 oraz R7, po wykonaniu działania inkrementują rejestr R0. W ten sposób następna wywołana procedura PLUS/MINUS pobierze daną z kolejnej komórki pamięci. Po wykonaniu działań na wszystkich ośmiu elementach danego wiersza, wywołana zostaje procedura ZAPIS. Jej zadaniem jest zapisanie wartości rejestrów R6 oraz R7. Rejestr R6 zawierający mniej znaczący bajt zapisany zostaje pod adres wskazywany przez R1. Zawartość rejestru R7 zostaje zapisana pod adres wskazywany przez R1, po zwiększeniu o 10h. Na koniec procedury ZAPIS realizowane jest przygotowanie do działań na kolejnym wierszu, rejestr R1 jest inkrementowany tak, aby wynik kolejnego wiersza został zapisany do kolejnej pary komórek. Do rejestrów R6 oraz R7 jest wpisywana ponownie liczba 1020 a do rejestru R0 liczba 20h. Procedura ta kończy swoje działanie wywołaniem instrukcji powrotu. Program przechodzi do działań na kolejnych wierszach. Po wykonaniu transformacji wszystkich wierszy następuje powrót do początku programu.

Procedura PLUS (rys. 4) rozpoczyna działanie od wpisania liczby 0 do rejestru R3. Następnie do rejestru R6 przez akumulator dodawana jest komórka pamięci wskazywana przez rejestr R0, wynik wpisany zostaje do R6. Następnie do rejestru R7 przechowującego starszy bajt z pomocą akumulatora dodawany jest znacznik C oraz R3=0. Dodawanie znacznika C do R7 zastosowano w celu zapisania bitu przeniesionego w dodawaniu do R6. Na koniec procedury następuje inkrementacja rejestru R0 w celu pobierania danych z kolejnej komórki przez następną procedurę. Procedura MINUS realizuje podobne instrukcje z tą różnicą, że zamiast dodawać odejmuje.

```
Wpisz 0 do rejestru R3;
Wpisz do akumulatora A zawartość rejestru R6;
Do A dodaj zawartość komórki wskazywanej przez rejestr R0;
Wypisz wynik z A do R6;
Wpisz zawartość R7 do A;
Dodaj rejestr R3 oraz znacznik C do A;
Wpisz A do R7;
Inkrementruj R0;
Powrót z procedury;
```

Rys. 4. Lista kroków procedury PLUS
Fig. 4. List of steps of the PLUS procedure

Procedura ZAPIS (rys. 5) wywoływana jest, jako koniec działania na każdym z wierszy. Zapisuje wynik działania na wierszu (sumowanie oraz odejmowanie zmiennych) do komórki pamięci wskazywanej przez R1 (młodszy bajt z R6) oraz do komórki pamięci wskazywanej przez R1 zwiększonej o 10h (starszy bajt z R7). Od tego też procedura rozpoczyna swoje działanie. Następnie przygotowuje rejestry do działania na kolejnych wierszach. Do rejestrów R6 oraz R7 wpisana zostaje liczba 0x3FC, natomiast do R0 liczba 20h.

```
Wpisz do A rejestr R6;
Skopiuj do komórki wskazywanej przez R1 zawartość A;
Zwiększ R1 o 10(16);
Skopiuj do A rejestr R7;
Skopiuj do komórki wskazywanej przez R1 zawartość A;
Wyczyść wskaźnik C;
Odejmij 10(16) od R1;
Inkrementuj R1;
Wpisz do R6 liczbę FC(16);
Wpisz do R7 liczbę 3(16);
Wpisz do R0 liczbę 20(16);
Powrót z procedury;
```

Rys. 5. Lista kroków procedury ZAPIS
Fig. 5. List of steps of the ZAPIS procedure

4. Wnioski

Przedstawiony w [1] układ zajmuje bardzo małą objętość. Jego koszty wytworzenia są niewielkie. Można bez trudu zamontować go do już działających urządzeń. Największy problem stanowi wykonanie praktyczne urządzenia blokującego dostęp do maszyny ze względu na ich różnorodność. Do analizy sygnału mowy wykorzystano analizę sekwencyjnościową (Walsha) a nie częstotliwościową (Fouriera), co znacznie przyśpieszyło analizę w czasie rzeczywistym oraz uprościło konstrukcję urządzenia.

5. Literatura

[1] Syroka Z., Słomian D., Kusyk K.: Sposób analizy i rozpoznawania mowy oraz układ do analizy i rozpoznawania mowy. Zgłoszenie patentowe P.389450 z dnia 15.02.2010r.
[2] Kleijn W., Paliwal K.: Speech coding and synthesis. Elsevier Science B.V., Amsterdam 1995.
[3] Furui S., Sondhi M.: Advances in Speech Signal Processing. Marcel Dekker, Inc, New York 1992.
[4] Flanagan J.: Speech Analysis Synthesis and Perception, Springer-Verlag, New York 1972.
[5] Beauchamp K.: Applications of Walsh and Related Functions with an Introduction to Sequency Theory. Academic Press, London 1984.
[6] Tzafestas S.: Walsh functions in signal and systems analysis and design. Van Nostrand Reinhold Company Inc., New York 1985.
[7] Harmuth H.: Sequency Theory – Foundation and Applications. Academic Press, New York 1977.
[8] Elliot D., Rao K.: Fast Transform – Algorithms, Analyses, Applications. Academic Press, New York 1982.
[9] Drygajło A., Rumatowski K.: Analiza sekwencyjnościowa układów liniowych, PWN Warszawa 1990.
[10] Syroka Z.: Bezprzewodowy system diagnostyczny. Zgłoszenie patentowe P.382991 z dnia 25.07.2007.

otrzymano / received: 01.10.2010
przyjęto do druku / accepted: 03.09.2012 *artykuł recenzowany / revised paper*