Zestaw DSP controlSTICK w praktyce

W EP8/09, EP9/09 oraz EP10/09 zaprezentowaliśmy zestaw startowy controlSTICK z mikrokontrolerem sygnałowym TMS320F28027. Zamieszczone w artykułach informacje powinny być wystarczające do wkroczenia w świat cyfrowego przetwarzania sygnałów z narzędziem w postaci zestawu controlSTICK. Teraz proponujemy niebanalny sposób pracy z zestawem controlSTICK, który umożliwia testowanie algorytmów DSP w praktyce.

Dodatkowe materiały na CD i FTP:
ftp://ep.com.pl, user: 18366, pass: 3scpp470
- listingi
- aplikacja sterująca Piccolo
- artykuły opisujące zestaw startowy controlSTICK z numerów EP8-10/2009

Cyfrowe przetwarzanie sygnałów jest dziedziną nauki trudną, a miejscami nawet bardzo trudną. Jednak samo wykorzystanie algorytmów DSP nie jest już aż tak arcytrudnym zadaniem. Tak jak operator dźwigu nie musi znać szczegółów funkcjonowania jego napędów, tak inżynier korzystający z DSP nie musi dokładnie rozumieć wszystkich zawiłości matematycznych zastosowanych algorytmów. Innymi słowy, jest możliwe korzystanie z algorytmów DSP przy minimalnej znajomości aparatu matematycznego.

Prezentowane podejście do zagadnień związanych z DSP zostało wykorzystane w niniejszym artykule, który koncentruje się na praktycznych aspektach zastosowania cyfrowego przetwarzania sygnałów. Praktycznych, ponieważ poznawane metody i poruszane problemy będą od razu prezentowane na platformie eksperymentalnej, której zadaniem jest wizualizacja wyników przetwarzania informacji przez procesor sygnałowy. Oczywiście podane informacje należy traktować jako wstęp do dalszej nauki, ponieważ celem artykułu jest demonstracja zastosowania DSP. Algorytmy będą traktowane jak rodzaj czarnej skrzynki, nie będą natomiast stanowić meritum opisywanych zagadnień.

Należy podkreślić, że o ile nie będziemy wnikać w zawiłości matematyczne, to niekiedy szczegółowo zostaną omówione niektóre elementy ich implementacji w układzie TMS320F28027. Zakłada się ponadto, że Czytelnikowi nie są obce takie podstawowe pojęcia związane z przetwarzaniem sygnałów, jak: odpowiedź impulsowa, odpowiedź skokowa, splot itd.

Platforma eksperymentalna

Z użyciem zestawu controlSTICK i komputera PC można zbudować niebanalne stanowisko do pracy z cyfrowym przetwarzaniem sygnałów. Na **rys. 1** przedstawiono blokowy schemat takiej platformy. Nie ma potrzeby dołączania jakichkolwiek zewnętrznych układów do płytki zestawu. Możliwości przedstawianej platformy eksperymentalnej nie są może oszałamiające, lecz wystarczają podczas pierwszych wypraw w świat DSP.

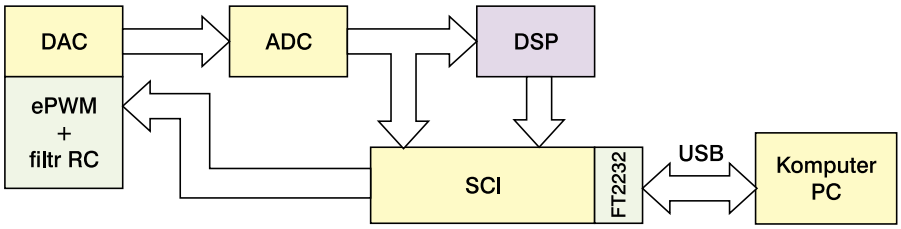
Sposób realizacji platformy eksperymentalnej jest dość oryginalny. Głównym założeniem było stworzenie narzędzi w możliwie krótkim czasie, aby móc przystąpić jak najszybciej do realizacji właściwych zadań. Komputerowa aplikacja sterująca i wizualizująca wyniki została zbudowana jako arkusz kalkulacyjny w MS Excel. Obecnie każdy użytkownik komputera posiada zainstalowany podstawowy pakiet MS Office lub jego odpowiednik, zatem nie powinno być najmniejszych problemów z uruchomieniem platformy eksperymentalnej.

Przedstawiane rozwiązanie charakteryzuje elastyczność, przy niemal całkowitej automatyzacji generowania wektora testowanego, pomiarów oraz wizualizacji. Nie potrzeba żadnych dodatkowych narzędzi czy sterowników. Wszystko, co jest niezbędne, to arkusz kalkulacyjny, zestaw controlSTICK zaktualizowany do obsługi wirtualnego portu szeregowego (opisywaliśmy to w EP 10/09) oraz pakiet Code Composer Studio.

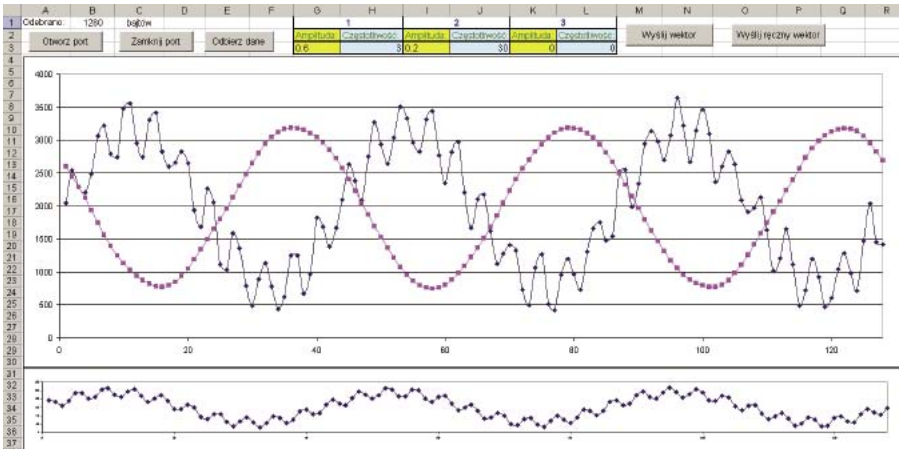
Zarówno wysyłany wektor, jak i odbierane dane mają długość 128 próbek. Można oczywiście długości wektorów zwiększać, jednak do pierwszych prób 128 próbek wydaje się wystarczające.

Aplikacja PC

Jak wspomniano wcześniej, aplikację służącą do sterowania mikrokontrolerem sygnałowym i wizualizacji wyników przygotowano w formie skoroszytu arkusza kalkulacyjnego



Rys. 1.



Rys. 2.

List. 1.
```
// Funkcja obslugi przerwania od timera 0
interrupt void timer0Isr(void)
{
        // Pobranie probki z ADC:
        probki_z_ADC[indeks] = AdcResult.ADCRESULT0;

        // Obliczenia DSP:
        fir.input=probki_z_ADC[indeks];
        fir.calc(&fir);
        probki_po_DSP[indeks] = fir.output;

        // Zmiana wsp. wyp. sygnalu PWM:
        EPwm1Regs.CMPA.half.CMPA = tab_wsp_wyp[indeks++];
        // Szybkie ,dzielenie' modulo:
        indeks &= 127;

        CpuTimer0Regs.TCR.bit.TIF = 1;
        // Grupa 1 „wolna":
        PieCtrlRegs.PIEACK.bit.ACK1  = 1;
}
```

MS Excel. Wygląd najważniejszych elementów skoroszytu przedstawiono na **rys. 2**. Przed przystąpieniem do pracy należy w edytorze VBA, w funkcjach obsługi każdego przycisku, nadać zmiennej *intPortID* wartość taką, jaki jest numer wirtualnego portu szeregowego, do którego jest podłączony zestaw controlSTICK.

Sygnały, które mają być wysłane do zestawu startowego, można deklarować na dwa sposoby. Jeśli chcemy badać zachowanie systemu podczas przetwarzania sygnałów okresowych, to wystarczy wpisać amplitudy i częstotliwości trzech sinusoid, które po zsumowaniu dają próbki wysyłane następnie do układu TMS320F28027 – wysłanie nastąpi po naciśnięciu przycisku *Wyślij wektor*.

Drugi sposób przekazywania próbek do przetwarzania polega na ręcznym wpisywania poszczególnych wartości w kolumnie AA, a następnie kliknięciu przycisku *Wyślij ręczny wektor*. Ręczne wpisywanie poszczególnych wartości próbek daje między innymi możliwość zbadania odpowiedzi impulsowej lub skokowej systemu. Należy pamiętać, że wpisywane tutaj wartości są pośrednio wypełnieniem sygnału PWM, który następnie po dolnoprzepustowej filtracji RC jest próbkowany przez przetwornik A/C. W dalszej części artykułu ten problem będzie omówiony bardziej szczegółowo.

List. 2.
```
// Funkcja obslugi przerwania od czesci nadawczej SCIA
interrupt void sciaTxFifoIsr(void)
{
        // Wykonuj dopóki kolejka nie zapełniona i nie
        // koniec komunikatu:
        while(SciaRegs.SCIFFTX.bit.TXFFST < 4 &&
                *(wsk_komunikatu+i) != ,\0')
        {
                SciaRegs.SCITXBUF = *(wsk_komunikatu+i);
                i++;
        }
        // Jeśli koniec komunikatu, zeruj licznik ,i'
        // oraz wyłącz przerwanie od TX:
        if(*(wsk_komunikatu+i) == ,\0')
        {
                i = 0;
                PieCtrlRegs.PIEIER9.bit.INTx2 = 0;
        }
        // Czysc flage przerwanaia:
        SciaRegs.SCIFFTX.bit.TXFFINTCLR = 1;
        // Grupa 9 gotowa na następne przerwanie:
        PieCtrlRegs.PIEACK.all |= 0x100;

}

// Funkcja obslugi przerwania od czesci odbiorczej SCIA
interrupt void sciaRxFifoIsr(void)
{
        char ReceivedChar;
        // Odczytanie odebranego znaku z bufora odbiorczego:
        ReceivedChar = SciaRegs.SCIRXBUF.all;
        if(ReceivedChar == ,;')
                odebrano_wyp = 1;
        if(ReceivedChar == ,d')
                wysylaj = 1;
        if(odebrano_wyp == 1 && j < 3 && ReceivedChar != ,;')
                wsp_wyp[j++] = ReceivedChar;
        // Odebrano wsp. wyp. (trzy znaki), wiec zapis
        // jego wartosci do tablicy wspolczynnikow wyp.:
        if(j == 3)
        {
                odebrano_wyp = 0;
                j = 0;
                // Zapis odebranego wsp. wyp do tablicy
                tab_wsp_wyp[indeks_tab_wsp++] = (wsp_wyp[2]-'0')
                        + (wsp_wyp[1]-'0')*10 + (wsp_wyp[0]-'0')*
                        100 - 100;
                // Szybkie ,dzielenie' modulo
                indeks_tab_wsp &= 127;
        }
        // Czyszczenie flag:
        SciaRegs.SCIFFRX.bit.RXFFOVRCLR = 1;
        SciaRegs.SCIFFRX.bit.RXFFINTCLR = 1;
        // Grupa 9 „wolna"
        PieCtrlRegs.PIEACK.all |= 0x100;
}
```

Gdy wyjściowy ciąg próbek zostanie już wysłany do zestawu controlSTICK, to wtedy można odebrać wynik przetwarzania DSP przez kliknięcie przycisku *Odbierz dane*. Zostanie on przedstawiony na wykresie, który zajmuje największą część okna aplikacji.

Na wykresie umieszczane są dwa przebiegi – wynik bezpośrednio z przetwornika A/C oraz efekt działania algorytmu DSP. Na małym wykresie u dołu okna jest pokazywany zadawany sygnał wzorcowy.

Aplikacja po stronie TMS320F28027

Przedstawienie aplikacji po stronie układu TMS320F28027 zaczniemy od opisu tego, co kolejno dzieje się z wysyłanymi do płytki controlSTICK próbkami sygnału zadawanego. Mikrokontroler sygnałowy komunikuje się z komputerem przez port szeregowy SCIA za pomocą układu mostu FT2232D, dzięki czemu ostatecznie uzyskujemy połączenie za pomocą interfejsu USB, choć od strony programowej, zarówno po stronie komputera, jak i mikrokontrolera, używany jest port szeregowy.

Dane odebrane przez port SCI po zdekodowaniu są przekazywane w cyklicznych odstępach czasu jako sterujące wypełnieniem do układu ePWM, na którego wyjściu jest podłączony filtr dolnoprzepustowy. W efekcie otrzymuje się sygnał analogowy. Częstotliwość odcięcia zastosowanego filtru RC wynosi około 34 kHz. Dalej sygnał analogowy jest podawany na wejście przetwornika A/C (w tym celu należy zewrzeć piny 3 i 31 wyprowadzenia szpilkowego zestawu controlSTICK).

Dane po wykonaniu przetwarzania A/C odfiltrowanego sygnału PWM, już w formie cyfrowej, są archiwizowane w tablicy (wzór od porównań) oraz równolegle poddawane cyfrowej obróbce, a wyniki obliczeń są kopiowane do drugiej tablicy. W ten sposób otrzymywane są dwie 128-elementowe tablice z danymi: w jednej znajdują się próbki sygnału surowego, pochodzącego wprost z przetwornika A/C, a w drugiej tablicy mamy wynik działania algorytmu DSP. Po kliknięciu przycisku *Odbierz dane* w aplikacji sterującej obie tablice zostaną przesłane do komputera i przedstawione w formie wykresów.

Mamy zatem kompleksowe rozwiązanie – jedna niewielka płytka, która jest generatorem, przetwornikiem A/C i „procesorem sygnałowym". Ostatnią frazę celowo ujęto w cudzysłów, ponieważ mikrokontroler czasu rzeczywistego TMS320F27028 nie jest przedstawicielem typowego procesora

sygnałowego, ale jego mocno uproszczoną wersją.

Program przygotowany dla mikrokontrolera sygnałowego Piccolo jest wyraźnie podzielony na bloki. Najważniejsze z nich, oprócz konfiguracji układów peryferyjnych, to:

– funkcja obsługi przerwania od timera0, której zadaniem jest: pobranie przetworzonej wartości z przetwornika A/C, zmiana współczynnika wypełnienia sygnału PWM oraz, co najważniejsze z naszej perspektywy – wykonanie obliczeń DSP – funkcję obsługi przerwania od timera0 przedstawiono na **list. 1**,
– funkcje obsługi przerwania od części odbiorczej i nadawczej SCIA – są one odpowiedzialne za obieranie danych do przetworzenia oraz wysyłanie danych przetworzonych do komputera, kod obu funkcji zamieszczono na **list. 2**,
– główna pętla programu, która jest odpowiedzialna za wysyłanie danych do komputera PC, patrz **list. 3.**

Kompletny projekt, przygotowany w pakiecie Code Composer Studio, podobnie jak aplikacja sterująca w formie skoroszytu, są zamieszczone na płycie CD dołączonej do numeru oraz na serwerze FTP.

List. 3.
```
while(1)
{
        if(indeks == 0 && wysylaj == 1)
        {
                IER &= ~M_INT1;
                wysylaj = 0;
                for(n = 0; n < 128; n++)
                {
                        nb2str(bufor,    probki_z_ADC[j]);
                        // Czekamy, az zakonczy sie wysylanie
                        // poprzedniego komunikatu:
                        while(PieCtrlRegs.PIEIER9.bit.INTx2 == 1);
                        while(SciaRegs.SCIFFTX.bit.TXFFST != 0);
                        wyslij_komunikat(bufor);;
                }
                for(n = 0; n < 128; n++)
                {
                        nb2str(bufor,    probki_po_DSP[j]);
                        while(PieCtrlRegs.PIEIER9.bit.INTx2 == 1);
                        while(SciaRegs.SCIFFTX.bit.TXFFST != 0);
                        wyslij_komunikat(bufor);

                }
                IER |= M_INT1;
        }
}
```

List. 4.
```
EALLOW;  // Zezwolenie na dostep do rejetrow chronionych

// Przerwanie od timer0 obslugiwane przez timer0Isr()
PieVectTable.TINT0 = &timer0Isr;

EDIS;   // Wylaczenie dostepu do rejestrow chronionych

PieCtrlRegs.PIECTRL.bit.ENPIE = 1;   // Wlaczenie PIE
// Wlaczenie przerwania od timer0 (INT1.7):
PieCtrlRegs.PIEIER1.bit.INTx7=1;
IER |= 0x100;    // Wlacznie przerwan CPU
IER |= M_INT1;
EINT;   // Wlaczenie obslugi przerw. globalnych

// Inicjalizacja timera0, przerwanie co 10ms
// definicje mSecX sa w PeripheralHeaderIncludes.h
CpuTimer0Regs.TCR.bit.TIE = 1;
CpuTimer0Regs.PRD.all =  mSec10;
```

Timer 0

Mikrokontroler czasu rzeczywistego TMS320F28027 ma wbudowane trzy 32-bitowe timery. W przedstawianej aplikacji wykorzystywany jest tylko jeden – timer 0. Jego rola jest bardzo ważna, ponieważ to timer 0 jest odpowiedzialny za wszystkie operacje związane z DSP, czyli: zmiana współczynnika wypełnienia układu ePWM, odebranie wyniku pomiaru z przetwornika A/C oraz to, co nas najbardziej interesuje, obliczenie DSP.

Układ timera 0 jest konfigurowany do pracy jako źródło przerwania. Kod, którego wykonanie powoduje ustawienie timera 0 do generowania przerwania co 10 ms, przedstawiono na **list. 4**.

Komunikacja

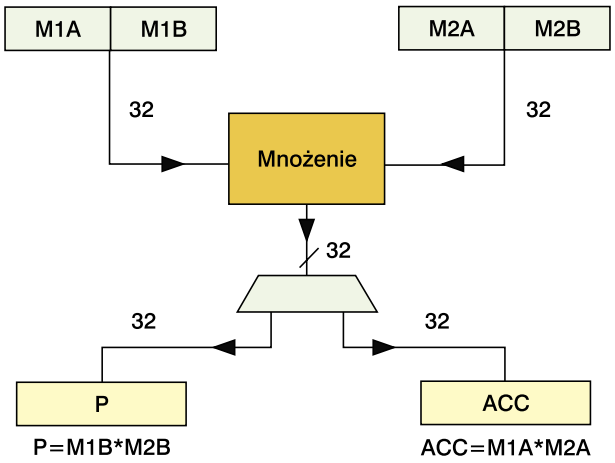
W trakcie normalnej pracy mikrokontroler czasu rzeczywistego w równych odstępach, wyznaczonych przez timer0 poza odczytywaniem wyniku przetwarzania A/C, zmiany współczynnika wypełnienia oraz obliczeń DSP, wypełnia również dwie tablice. W tablicy *probki_z_ADC[]* zapisywane są wartości próbek pochodzących wprost z przetwornika A/C, a w tablicy *probki_po_DSP[]* zapisywane są wyniki działania algorytmów DSP. Czas, jaki jest potrzebny na wykonanie obliczeń na całym zadanym sygnale wejściowym, jest wyznaczony przede wszystkim przez odstępy czasu, w jakich wywoływane jest przerwanie od timera0. Zgodnie z tym, co zostało wyżej napisane, długość przetwarzanego sygnału wynosi 128 próbek, a w związku z tym, że timer skonfigurowano do generowania przerwania co 10 ms, czas potrzebny na przetworzenie całego zadanego sygnału można wyznaczyć z prostej zależności: t=128×0,01 s=1,28 s. Dużo, jednak do zilustrowania przykładu dolnoprzepustowego filtru FIR wystarczy.

W układach TMS320F28027 zastosowano 12-bitowy przetwornik A/C. W związku z tym maksymalna liczba, którą można uzyskać z przetwornika, to 2^{12}, czyli 4096. Jeśli dane mają być przesyłane w formie znaków, to na każdą próbkę potrzeba minimum 4 bajty plus separator, na przykład w postaci średnika (abyśmy mogli łatwo odróżnić od siebie próbki), co daje 5 bajtów na pojedynczą próbkę. W sumie otrzymujemy 5 bajtów×128 próbek×2 (ponieważ są dwa wektory)=1280 bajtów. Dotyczy to oczywiście komunikacji w kierunku płytka controlSTICK –> komputer PC.

Sposób wysyłania i kodowania informacji przeznaczonej do wizualizacji na komputerze jest następujący. Główna pętla programu (czyli *while(1)*) sprawdza cały czas, czy komputer wysłał żądanie przesłania wyników pomiarów i przetwarzania DSP. Jeśli w buforze odbiorczym układu interfejsu szeregowego SCIA znajdzie się literka 'd', to jest ustawiana zmienna *wysylaj*, a co za tym idzie, rozpoczyna się wysyłanie danych do komputera. Dodatkowy warunek *indeks==0* sprawia, że dane przesyłane będą zawsze po wykonaniu obliczeń na wszystkich 128 próbkach. W pierwszej kolejności wysyłane są próbki z przetwornika A/C, czyli te niepoddane obróbce, a następnie wyniki działania algorytmu DSP.

Dane w obydwu tablicach są zapisane jako liczby 16-bitowe, więc przed rozpoczęciem wysyłania niezbędna jest konwersja każ-

List. 5.
```
// Konwersja liczby do tablicy znakow
void nb2str(char * buf, Uint16 nb)
{
        // Znak konca lancucha:
        buf[5] = ,\0';
        // Separator ,;'
        buf[4] = ,;';
        // Cyfra jednostek:
        buf[3] = nb % 10 + ,0';
        // Cyfra dziesiatek:
        buf[2] = (nb / 10) % 10 + ,0';
        // Cyfra setek:
        buf[1] = (nb / 100) % 10 + ,0';
        // Cyfra tysiecy:
        buf[0] = nb / 1000 + ,0';
}
```



Rys. 3.

dej próbki na ciąg znaków. Zajmuje się tym funkcja *nb2str()*, której ciało przedstawiono na **list. 5**. Wyłuskanie kolejnych cyfr odbywa się przez stosowanie kombinacji operacji dzielenia modulo i dzielenia całkowitego.

Podczas komunikacji w drugą stronę, czyli z komputera PC do płytki controlSTICK, przesyłane jest wypełnienie sygnału PWM, docelowo generowanego przez układ ePWM. Wypełnienie jest kodowane zawsze za pomocą trzech znaków plus separator w postaci średnika, co w sumie daje 512 bajtów przypadających na wektor sygnału zadawanego.

Generator przebiegu PWM jest taktowany systemowym sygnałem zegarowym SYSCLK, który wynosi 60 MHz, natomiast okres licznika PWM podczas konfiguracji ustawiono na 120. Częstotliwość generowanego sygnału PWM wynosi: 60 MHz/120=500 kHz.

Patrząc na funkcję obsługi przerwania od części odbiorczej SCIA na list. 2, widzimy że po obliczeniu wartości liczbowej z trzech cyfr w formacie ASCII odejmowana jest jeszcze liczba 100. Jest to uzasadnione tym, że do każdej wartości wypełnienia, jaka jest wysyłana przez komputerową aplikację sterującą, dodawane jest 100. Taki zabieg gwarantuje, że zawsze, nawet jeśli w arkuszu kalkulacyjnym wpisane zostanie wypełnienie jedno- lub dwucyfrowe, przesyłane będą trzy znaki na każdą próbkę wypełnienia.

Przykład – filtr FIR

Wykorzystanie oraz prezentacja części możliwości omawianej platformy eksperymentalnej zostaną przedstawione na przykładzie realizacji dolnoprzepustowego filtru o skończonej odpowiedzi impulsowej (*Finite Impulse Response – FIR*). Jest to jeden z najprostszych algorytmów DSP, a w ogólnym przypadku realizuje się go za pomocą operacji splotu sygnału oraz odpowiedzi impulsowej filtru. Załóżmy, że odpowiedź impulsowa filtru *h(n)* jest nadzwyczaj krótka, o długości czterech próbek. Splot w tym przypadku dla próbki wyjściowej *y(5)* będzie wyglądał następująco (*x(n)* jest sygnałem wejściowym):

$$y(5) = h(0)x(5) + h(1)x(4) + h(2)x(3) + h(3)x(2)$$

Przedstawiając powyższe równanie słownie że można napisać, ze splot dla danej próbki wyjściowej to suma iloczynów historycznych, próbek wejściowych oraz próbki aktualnej, czyli tego, co było i jest na wejściu z odpowiadającymi im współczynnikami filtru FIR (odpowiedzi impulsowej).

Odpowiedź impulsowa, podobnie jak odpowiedź częstotliwościowa oraz skokowa (ta ostatnia jest całką odpowiedzi impulsowej), daje pełną informację o zachowaniu systemu na dowolny sygnał wejściowy. Kształtując odpowiedź impulsową, decydujemy o tym, CZYM będzie projektowany system DSP oraz jak się będzie ZACHOWYWAŁ w reakcji na sygnał wejściowy.

W omawianej aplikacji wykorzystano biblioteki firmy TI, do której współczynniki filtrów 50. rzędu są dołączone jako przykład. Dysponując odpowiednimi narzędziami można przystąpić do badania właściwości naszego skromnego systemu DSP. W pierwszej kolejności zbadane zostanie, czy system w ogóle działa. Innymi słowy, należy sprawdzić, czy zadane przebiegi są generowane oraz czy filtr działa zgodnie z intuicją.

Po załadowaniu programu do pamięci i uruchomieniu mikrokontrolera (patrz EP08/09), jeśli tylko w Excelu w edytorze VBA mamy wpisany w funkcjach obsługi przycisków numer portu, pod którym dostępna jest płytka controlSTICK, możemy przystąpić do pracy.

Niech badany sygnał składa się z dwóch sinusoid o częstotliwościach 3 i 30. Warto podkreślić, że częstotliwości są wyrażane w wartościach względnych, w rzeczywistości częstotliwość generowanego sygnału jest wyznaczana przez odstępy czasu, po jakich wywoływane jest przerwanie od timera 0.

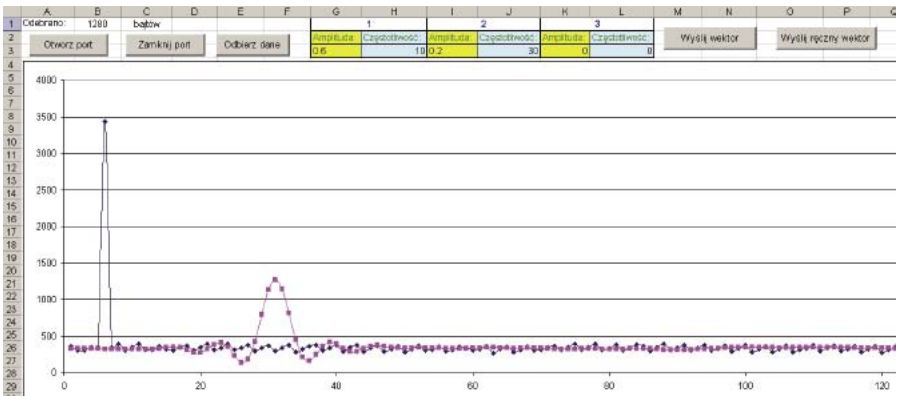
Amplituda jest znormalizowana, a więc wartości, jakie są wpisywane, powinny być większe od 0 i mniejsze od 1. W przykładowym sygnale amplitudy składowych wynoszą 0,6 oraz 0,2.

Gdy sygnał mamy już zdefiniowany, otwieramy port, wysyłamy wektor, a następnie odbieramy dane z płytki zestawu startowego za pomocą przycisku *Odbierz dane*. Efekt, jaki zostanie uzyskany, powinien być podobny do tego z rys. 2.

W kolejnym etapie zbadana zostanie odpowiedź impulsowa filtru. W tym celu należy stworzyć sygnał, który zostanie wysłany do obróbki. Definiowanie sygnałów odbywa się przez edytowanie współczynnika wypełnienia dla PWM w kolumnie AA aplikacji sterującej. Odpowiedź impulsowa, jak jej nazwa wskazuje, jest tylko jednym impulsem, a więc ustawiamy wypełnienie, na przykład dla piątej próbki, na 120. Jest to maksymalna wartość wypełnienia, która została ustalona wraz z deklarowaniem okresu licznika ePWM. Pozostałe wartości próbek niech wynoszą 10. Po załadowaniu wektora do mikrokontrolera sygnałowego i odebraniu danych w aplikacji sterującej powinna się pojawić odpowiedź impulsowa systemu, taka jaką pokazano na **rys. 4**.

W systemach, w których informacja zawarta jest w kształcie sygnału, parametrem najlepiej ukazującym właściwości systemu jest odpowiedź skokowa. Sprawdźmy więc, jaką odpowiedź skokową ma nasz filtr. Podobnie jak w poprzednim przypadku, w pierwszym etapie należy przygotować sygnał pobudzający. Niech wektor testowy będzie następujący: pierwsze cztery próbki o wartości 10, następne 50 próbek o wartości 100, a reszta próbek niech ma wartość 10.

Po wysłaniu, a następnie odebraniu danych, na wykresie w aplikacji sterującej ukaże się symetryczna odpowiedź skokowa,



Rys. 4.

Artykuły na temat mikrokontrolerów czasu rzeczywistego Piccolo – TMS320F28027, które dotychczas ukazały się na łamach EP:
EP08/09 – Mocarny Maluch. Piccolo – najmniejszy DSP z Texasa
EP09/09 – DSP w praktyce. Konfiguracja przetwornika A/C w TMS320F28027
EP10/09 – DSP w praktyce. Generator PWM i interfejs SCI

którą przedstawiono na **rys. 5**. Symetryczna, ponieważ wysłany został sygnał skokowy ze zboczem narastającym oraz opadającym.

Instrukcja DMAC

Algorytmy przetwarzania sygnałów charakteryzują się tym, że najczęściej pojawiającą się operacją arytmetyczną jest sekwencja „mnożenie-dodawanie-akumulacja" (MAC – *multiply-add-accumulate*), czyli tzw. suma splotowa, co zapisujemy jako:

$$y(n) = \sum_{k=a}^{b} h(k)x(n-k)$$

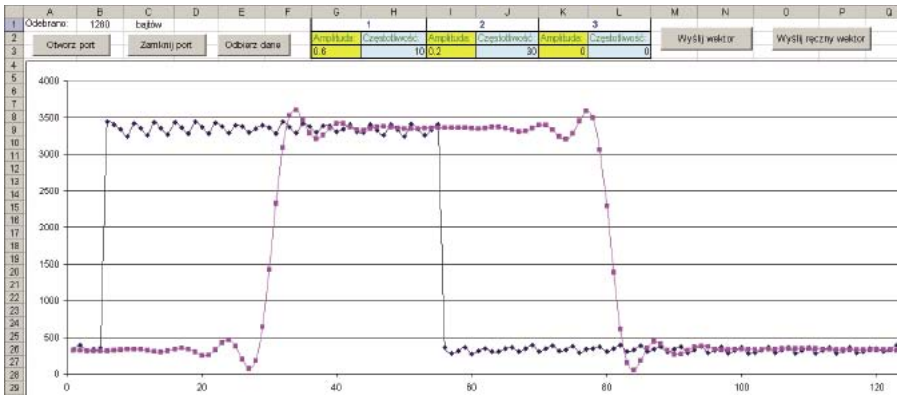
Z tego powodu rdzenie procesorów sygnałowych mają zazwyczaj wbudowane jednostki sprzętowo wspomagające operacje mnożenia MAC. Rdzeń C28x umożliwia sprzętowe mnożenie liczb 16-bitowych (16-bit X 16-bit MAC) oraz 32-bitowych (32-bit X 32-bit MAC). Dodatkową funkcjonalnością jest instrukcja DMAC, która pozwala na wykonanie dwóch mnożeń liczb 16-bitowych jednocześnie. Skupimy się na tej ostatniej ze względu na to, że to ona została wykorzystana w bibliotekach firmy TI do realizacji filtru FIR.

Sposób wykonywania podwójnego, jednoczesnego mnożenia przedstawiono na rys. 3. Mnożne dla obydwu mnożeń są przekazywane do jednostki mnożącej jako jedna wartość 32-bitowa (analogicznie mnożniki). Starsze 16-bitów (słowa) z obydwu przekazywanych liczb są traktowane jako jedno działanie, natomiast młodsze słowa z przekazywanych wartości są czynnikami drugiego działania mnożenia. Po wykonaniu operacji iloczyn z mnożenia starszych słów znajduje się w akumulatorze ACC, a ściślej wynik mnożenia jest dodawany do aktualnej wartości w akumulatorze (MAC – *multiply-add-accumulate*). Analogicznie wynik operacji MAC na młodszych słowach (16 bitach) jest akumulowany w rejestrze produktu P.

Bufor cykliczny

Obok MAC jedną z najważniejszych operacji w DSP jest buforowanie cykliczne (kołowe). Architektury procesorów sygnałowych są często optymalizowane pod kątem wykorzystania buforów cyklicznych. Celem stosowania buforów kołowych jest szybkie operowanie danymi przeznaczonymi do przetwarzania przy relatywnie niskim zużyciu zasobów. Bufor cykliczny jest po prostu miejscem w pamięci, które ma określony początek i koniec. Ponadto każdy bufor kołowy ma przynajmniej jeden wskaźnik na dane wewnątrz bufora. W przypadku najprostszym jest to jeden wskaźnik, który zawsze wskazuje najnowszą próbkę.

W mikrokontrolerach sygnałowych Piccolo buforowanie cykliczne, tak jak w prawdziwych procesorach sygnałowych, jest wspierane sprzętowo. Bufor kołowy jest realizowany przez specjalny tryb cyklicznego adresowania bezpośredniego. Wykorzystuje się do tego celu dwa rejestry procesora: XAR1 i XAR6. Długość bufora kołowego jest wyznaczona przez zawartość rejestru XAR1, natomiast rejestr XAR6 jest wskaźnikiem na miejsce w pamięci, gdzie znajduje się najnowsza próbka. Realizacja bufora kołowego w instrukcji asemblerowej jest osiągana przez zapis: *XAR6%++. Podobnie jak w Języku C, gwiazdka '*' oznacza, że zawartość rejestru jest adresem pamięci, procent '%' informuje natomiast o zastosowaniu adresowania cyklicznego. Inkrementacja adresu w rejestrze XAR6, również analogicznie do języka C, jest wykonywana przez '++'. Gdy najmłodsze bajty rejestrów XAR1 oraz XAR6 są sobie równe, wtedy następuje wyzerowanie najmłodszego bajta rejestru XAR6, dzięki czemu zachowana jest ciągłość w działaniu bufora kołowego.



Rys. 5.

DMAC i bufor cykliczny

Instrukcja DMAC i sprzętowo wspierane buforowanie cykliczne w połączeniu dają bardzo wydajne narzędzie dla cyfrowego przetwarzania sygnałów. Po wcześniejszym przygotowaniu wszystkich rejestrów realizacja splotu próbek wejściowych z odpowiedzią impulsową systemu może być zrealizowana za pomocą tylko dwóch linii kodu w asemblerze:

```
RPT     AR0
DMAC    ACC:P,*XAR6%++,*XAR7++
```

Linia kodu po instrukcji RPT będzie wykonywana tyle razy, ile wynosi zawartość rejestru AR0, który jest młodszym słowem rejestru XAR0. Rejestr XAR6 jest, jak wcześniej napisano, wskaźnikiem bufora kołowego, natomiast rejestr XAR7 to wskaźnik na tablice współczynników filtru. Trudno sobie wyobrazić bardziej zwięzły kod wykonujący operację splotu. Biorąc pod uwagę fakt, że jest to instrukcja asemblerowa, można być naprawdę pod wrażeniem. Kompletny kod realizujący filtrację pochodzi z biblioteki firmy TI i jest dołączony do projektu w pliku fir16.asm.

Podsumowanie

Tak jak żaden początkujący żeglarz nie wyrusza od razu w podróż dookoła świata, tak nikt nie rozpoczyna nauki DSP od przetwarzania obrazu wideo HD w czasie rzeczywistym. Przedstawiony w artykule sposób nauki nie jest rzecz jasna optymalny, jednak pozwala od razu zetknąć się z algorytmami DSP w praktyce. Jest to bardzo istotne, ponieważ często się zdarza, że o ile algorytmy cyfrowego przetwarzania sygnałów działają bardzo dobrze na dużych komputerach z ogromną ilością zasobów w wyspecjalizowanych programach, to w małych, nie do końca sygnałowych procesorach szybko kończą się dostępne zasoby, a optymalizacja działania jest silnie związana z architekturą danego układu.

Krzysztof Paprocki
paprocki.krzysztof@gmail.com