Język Verilog w przykładach (5)

Generowanie sygnału sinusoidalnego oraz sterowanie przetwornikiem C/A

W tym odcinku pokażemy, jak można wygenerować przebieg sinusoidalny korzystając tylko z zasobów cyfrowych FPGA. Opisany przykładowy układ może być użyty do budowy rekursywnych filtrów cyfrowych. Na przykładzie komunikacji z przetwornikiem cyfrowo-analogowym zademonstrujemy również, jak przeprowadza się transmisję za pomocą interfejsu SPI.

Jest wiele metod syntezowania sinusoidy, np. z użyciem tablicy LUT (*look-up-table*) czy wielomianów interpolacyjnych. Jedna z najprostszych polega na zbudowaniu rezonatora harmonicznego o zadanej częstotliwości rezonansowej i wzmocnieniu równym 1, w oparciu o rekursywny filtr cyfrowy IIR (*Infinite Impulse Response*). Jest ona bardzo łatwa do zrealizowania zarówno za pomocą procesorów jak i programowalnych struktur typu FPGA. Niestety, metoda ta ma również kilka poważnych wad. Ponieważ dla zmiany częstotliwości pracy należy zmienić współczynniki filtru i wpisać nowy plik konfigurujący do FPGA, to nie jest możliwa płynna zmiana częstotliwości (praktycznie zawsze nastąpi skok w fazie). Ponadto, aby ustawić wartości początkowe oraz parametry filtru tak, aby uzyskać generator o zadanej częstotliwości, to niezbędne są dodatkowe obliczenia. Wartości te można zgrupować w pamięci lub przesyłać je na bieżąco, np. z komputera. Pomimo tych ograniczeń rezonator harmoniczny jest wykorzystywany dość często, a przy okazji jest dobrym przykładem konstrukcji filtru cyfrowego w FPGA.

Aby uzyskać rezonator harmoniczny, transmitancja filtru powinna być drugiego rzędu, a więc mieć dwa bieguny i co najwyżej dwa zera. Transmitancja taka zwana jest w literaturze anglojęzycznej jako *biquad:*

$$H_{biquad}(z) = \frac{Y(z)}{X(z)} = \frac{a_0+a_1z^{-1}+a_2z^{-2}}{b_1z^{-1}+b_2z^{-2}}$$

przekształćmy to wyrażenie do postaci rekurencyjnej:

$$\begin{aligned} y[n] &= a_0x[n] + a_1x[n-1] + a_2x[n-2] \\ &\quad - b_1y[n-1] - b_2y[n-2] \\ &= \sum_{i=0}^{2} a_ix[n-i] - \sum_{j=1}^{2} b_jx[n-j] \end{aligned}$$

Jest to opis struktury filtru w formie bezpośredniej (*direct*). Jest ona przedstawiona na **rys. 5.1a**. Jest to często stosowana postać filtru w torach przetwarzania. Jej główną zaletą jest uniwersalność. Możliwa jest bowiem implementacja w tej postaci każdego rodzaju filtru (dolno- i górnoprzepustowego, pasmowoprzepustowego i pasmowozaporowego oraz właśnie rezonatorów itp.). Oczywiście filtry te mogą być maksymalnie drugiego rzędu, ale w razie potrzeby można je łączyć kaskadowo, otrzymując filtr dowolnego rzędu. Filtry drugiego rzędu są również stosunkowo łatwe w implementacji dzięki ich dość dobrej stabilności numerycznej.

Przedstawiony na rys. 5.1a schemat blokowy filtru wynika bezpośrednio ze wzoru rekurencyjnego. W literaturze i praktycznych implementacjach spotkać można zmodyfikowaną jej wersję, zoptymalizowaną pod kątem wykorzystania zasobów docelowej architektury sprzętowej. Dzięki odpowiednim przekształceniom strukturę filtru z rys. 5.1a można zredukować do postaci nie z dwoma, ale jedną linię opóźniającą (**rys. 5.1b**).

W tym przypadku nie będziemy korzystać z zer transmitancji – rezonator harmoniczny zrealizujemy umieszczając dwa sprzężone bieguny blisko krawędzi okręgu jednostkowego na płaszczyźnie zespolonej, przy czym moduł transmitancji (wzmocnienie) musi być równy jedności, aby rezonator działał prawidłowo. Nasz filtr będzie zatem opisywany równaniem:

$$y[n] = -b_1y[n-1] - b_2y[n-2] + x[n]$$

Aby skonfigurować rezonator potrzebne są cztery wartości: dwie próbki startowe oraz dwa współczynniki filtru. Dwie pierwsze wartości określają stan początkowy linii opóźniającej. Wejście jest w istocie nieużywane, a więc przyjmuje wartość 0, dlatego w kolejnych wyrażeniach czynnik x(n) nie występuje.

Jak znaleźć te parametry? Załóżmy, że chcemy aby filtr miał znormalizowaną pulsację rezonansową:

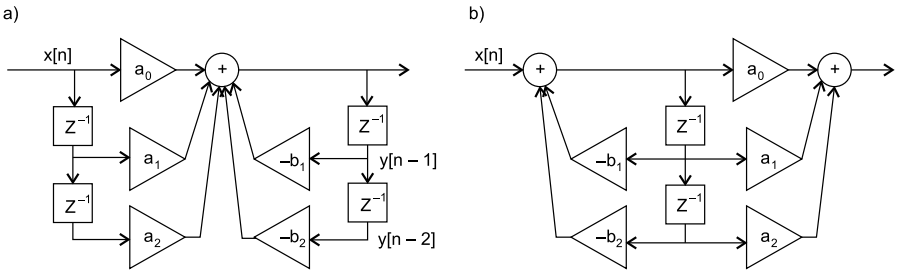
$$\frac{\omega}{\omega_p} = 2\pi\frac{f}{f_p} = 2\pi\frac{1}{40}$$

Przykładowo, dla częstotliwości próbkowania *fp=40 kHz* częstotliwość sygnału byłaby równa 1 kHz. Przyjmujemy zatem równanie opisujące sinusoidę z amplitudą również równą 1:

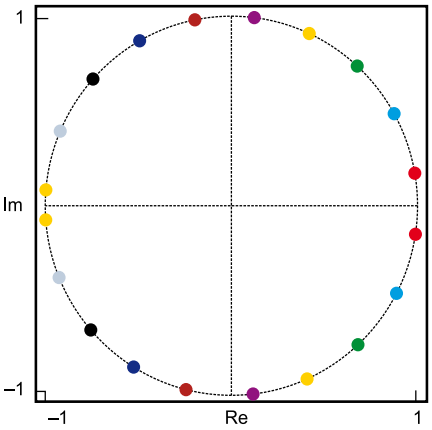
$$y[n] = \sin\left(2\pi\frac{1}{40}n\right)$$

a więc: y[0]=0, y[1]=0,1564, y[2]=0,3090, y[3]=0,4540.

Należy zaznaczyć, że dobór dwóch próbek startowych umożliwia regulację amplitudy drgań. Aby zatem otrzymać amplitudę drgań równą 0,9, należy rozwiązać poprzednie równanie przyjmując taką właśnie amplitudę. Co ważne, parametry filtru, które zostaną obliczone, nie zmieniają swojej wartości. Sposób ten pozwala na uniknięcie zastosowania układu mnożenia (multiplikatora) na wyjściu rezonatora w celu przeskalowania wartości amplitudy. Współczynniki filtru można obliczyć podstawiając wszystkie cztery obliczone próbki do układu równań, a następnie rozwiązując go względem współczynników *b1* i *b2*:



Rys. 5.1 Schemat blokowy rekursywnych filtrów cyfrowych IIR drugiego rzędu: a) postać bezpośrednia, b) postać przekształcona



Rys. 5.2 Rozmieszczenie par biegunów w przykładowych rezonatorach

$$\begin{cases} y[2] = -b_1 y[1] - b_2 y[0] \\ y[3] = -b_1 y[2] - b_2 y[1] \end{cases}$$

W tym przypadku współczynniki mają następujące wartości: $b1 = -1{,}9754$, $b2 = 1$.

Podstawmy do równania filtru obliczone wartości współczynników:

$$y[n] = 1.9754\, y[n-1] - y[n-2] + x[n]$$

Porządkując je ze względu na zmienne otrzymamy:

$$y[n] - 1.9754\, y[n-1] + y[n-2] = x[n]$$

Stosując transformację „z" otrzymamy transmitancję:

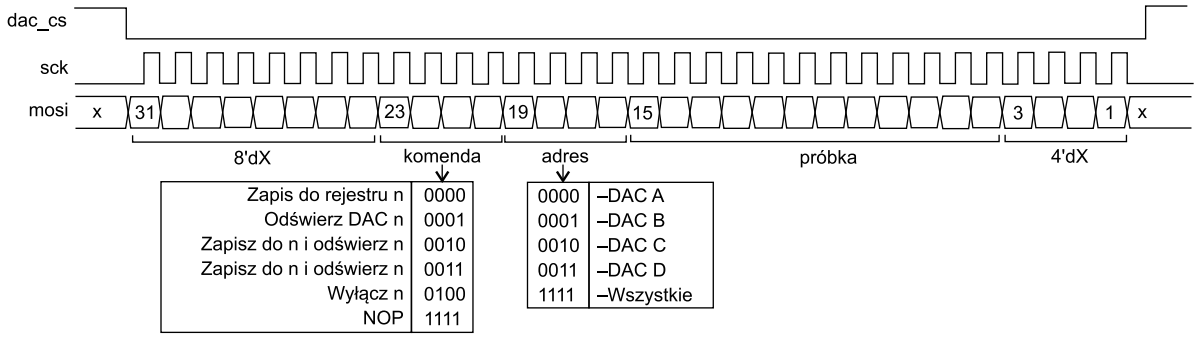
$$\frac{y(z)}{x(z)} = \frac{1}{1 - 1.9754 z^{-1} + z^{-2}} = \frac{1}{(1 - (0.9877 + j0.1564)z^{-1})(1 - (0.9877 - j0.1564)z^{-1})}$$

Widać, że bieguny są sprzężone i znajdują się na krawędzi okręgu jednostkowego na płaszczyźnie zespolonej (**rys. 5.2** – kolor czerwony). Wzmocnienie tego układu jest równe dokładnie 1. Gdyby znalazły się one poza okręgiem, wzmocnienie byłoby większe od 1 i układ stałby się niestabilny (amplituda generowanych oscylacji wzrastałaby). Gdyby bieguny zlokalizowane byłyby bliżej środka okręgu, wzmocnienie byłoby mniejsze od jedności, to generowane drgania byłyby tłumione i prędzej czy później zanikłyby całkowicie. Dlatego tak ważne jest, aby obliczenia przeprowadzać z dużą precyzją, zaokrąglając wartości dopiero na poziomie przekształcania współczynników do postaci stałoprzecinkowej, bo w takiej formie będziemy przeprowadzać obliczenia w FPGA. Na rys. 5.2 zebrane zostało 10 przykładowych par biegunów obliczonych dla zakresu unormowanych częstotliwości (0,025, 0,49).

Skoro precyzja wykonywania obliczeń jest krytyczna dla odpowiedniego działania rezonatora, to pojawia się problem reprezentacji liczb rzeczywistych w postaci stałoprzecinkowej oraz wykonywania na nich obliczeń. Przyjęto 12-bitową arytmetykę zmiennoprzecinkową, przy czym 11 bitów mniej znaczących reprezentuje część ułamkową, a 1 bit (MSB) część całkowitą liczby. Taki zapis zwany jest kodem Q11 (1.11). Zapewnia on reprezentację liczb ułamkowych do 4 miejsca po przecinku oraz liczb całkowitych do 1, co w tym przypadku powinno być wystarczające – wartość wynikowa nigdy nie powinna być większa od 1 (przy założeniu, że wartości startowe będą odpowiednio dobrane, a więc amplituda obliczanej sinusoidy nie będzie większa od 1). Odsyłam do literatury [3], ponieważ zagadnienie doboru formatu liczb stałoprzecinkowych do danej aplikacji jest skomplikowanym zagadnieniem, które znacznie wykracza poza zakres tematyczny tego artykułu. Konwersja liczb w postaci ułamkowej na wspomniany kod dokonywana jest przez pomnożenie ich przez wartość i oczywiście obcięcie części ułamkowej wyniku.

List. 5.1. Kod źródłowy rezonatora harmonicznego

```
module sine_iir
        # (parameter dlength=12)
        (clk, clrn, y, sample);
        //moduł rezonatora harmonicznego
        input clk, clrn, sample;
        output signed [dlength-1:0] y;

        reg signed [dlength-1:0] y_mem [1:0];
        //przeliczenia parametrów na
        //kod dopełnienia do 2 Q11 (1.11)
        //wartości startowe dla sinusoidy
        //o amplitudzie 0.9
        //i częstotliwości unormowanej: 1/40
        // 0*2^11=0
        // 0.1564 * 2^11 = 288
        //współczynniki filtra:
        //1.94754 * 2^11 = 4045.6 = 4046
        //1*2^11 = 2048

        assign y = y_mem[1];
        always@(posedge clk or negedge clrn)
                if (!clrn) begin
                //ładowanie wartości startowych
                        y_mem[0] <= 0;
                        y_mem[1] <= $signed(32'd288);
                end else if (sample) begin
                //działanie filtra
                        y_mem[0] <= y_mem[1];
                        y_mem[1] <= ($signed(32'd4046)*y_mem[1]>>>11)-
($signed(32'd2048)*y_mem[0]>>>11);
                end
        endmodule
```

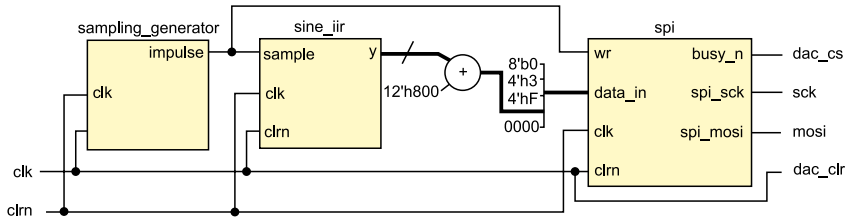


Rys. 5.3 Ramka SPI dla przetwornika LTC2624

Na **list. 5.1** przedstawiono kod źródłowy opisywanego rezonatora harmonicznego. Zwróćmy uwagę na sposób operowania współczynnikami filtru oraz na opis rejestru opóźniającego *y_mem*. W opisie rejestru jest dyrektywa *signed*. Jest to sposób poinformowania programu syntezy, że dana stała czy rejestr powinna być traktowana jako liczba ze znakiem w kodzie uzupełnienia do dwóch. Należy pamiętać, że stosując liczby ze znakiem, dla stałych należy użyć operatora *$signed(.)*, w szczególności przy operacjach arytmetycznych. Dzięki temu program syntezy zrealizuje operację ze znakiem i wynik będzie prawidłowy. O ile w przypadku dodawania i odejmowania nie ma problemów, to przy mnożeniu należy pamiętać, że mnożąc przez siebie dwie liczby 12-bitowe, wynik będzie reprezentowany przez liczbę 24-bitową w kodzie Q36 (2.22) – pozycja przecinka zostanie przesunięta na bit 22. Aby powrócić do 12-bitowej reprezentacji w kodzie Q11 należy wynik podzielić przez , a więc przesunąć o 11 pozycji w prawo i wybrać 12 mniej znaczących bitów wyniku.

Aby umożliwić kontrolę częstotliwości próbkowania generowanego sygnału, opisany został sygnał *sample*. Jego poziom wysoki umożliwia przeprowadzenie kolejnego kroku obliczeniowego. Aby zatem wytworzyć sinusoidę o częstotliwości np. 40 kHz, należy doprowadzić do tego wejścia sygnał impulsowy o tej częstotliwości, przy czym długość impulsu powinna być równa jednemu cyklowi zegarowemu.

Spróbujmy wyprowadzić wygenerowany sygnał sinusoidalny na zewnątrz FPGA. Najczęściej na płytkach uruchomieniowych są dostępne przetworniki cyfrowo-analogowe z interfejsem SPI (*serial peripheral interface*). Przykładowo wysterujemy czterokanałowy, 12-bitowy przetwornik C/A (LTC2624 firmy Linear Technology) zainstalowany na płycie uruchomieniowej *Spartan-3E Starter Kit HW-SPAR3E-SK-UK*. Komunikacja z tym układem odbywa się za pośrednictwem standardowego interfejsu SPI oraz dwóch dodatkowych sygnałów: *DAC_CS* oraz *DAC_CLR*. *DAC_CS* jest sygnałem *chip-select* o aktywnym poziomie niskim, przy czym konwersja cyfrowo-analogowa rozpoczyna się dopiero po powrocie tej linii na poziom wysoki. *DAC_CLR* jest asynchronicznym sygnałem resetu przetwornika, który podłączymy do wyprowadzenia sygnału resetu układu FPGA. Obliczenia w module *sine_iir* były wykonywane na liczbach w kodzie uzupełnienia do dwóch, a więc jednej z reprezentacji stałoprzecinkowych umożliwiającej zapis liczb ze znakiem. Przetwornik z kolei interpretuje dane wejściowe jako liczbę bez znaku – daje to nieprawidłowy wynik na jego wyjściu. Aby spowodować prawidłowe wygenerowanie sinusoidy, należy dodać do próbki wyjściowej z modułu *sine_iir* liczbę równą połowie zakresu rozdzielczości bez znaku (tu: 12'h8000, a więc dziesiętnie 2048).

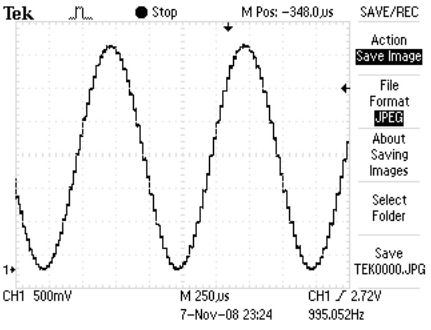


Rys. 5.4 Schemat układu zaprojektowanego generatora wraz z interfejsem SPI

List. 5.2 Opis źródłowy modułu transmisji SPI

```
module spi
        #(parameter dlength=32, parameter cnt_length=5)
        (input [dlength-1:0] data_in,
        input clk,
        input wr,
        output busy_n,
        input clrn,
        output spi_sck,
        output spi_mosi
        );
        //moduł transmisji szeregowej SPI
        //umożliwia przesłanie dowolnie długiej ramki
        //częstotliwośc zegara transmisji jest równa
        //częstotliwości zegara taktującego moduł

        reg [dlength-1:0] sreg_out; //rejestr przesuwny
        reg [cnt_length-1:0] cnt; //licznik bitów
        reg busy_int;

        assign busy_n = !(busy_int || wr);
        assign spi_sck = (!busy_int)?1'b0:~clk; //zegar transmisji
        assign spi_mosi = sreg_out[dlength-1]; //linia danych

        always@(posedge clk or negedge clrn)
                if (!clrn) begin
                        sreg_out <= {dlength{1'b0}};
                        busy_int <= 1'b0;
                        cnt      <= {cnt_length{1'b0}};
                end else if (!busy_int) begin
                        if (wr) begin
                                sreg_out <= data_in; //wpis danych do
transmisji
                                cnt <= {dlength{1'b0}}; //inicjalizacja
                                busy_int <= 1'b1;
                        end
                end else begin
                        cnt <= cnt +1;   //inkrementacja licznika bitów
                        //przesuwanie danych w rejestrze
                        sreg_out <= {sreg_out[dlength-2:0], 1'bx};
                        if (cnt==dlength-1) //koniec transmisji
                                busy_int <= 1'b0;
                end

endmodule
```

Układ przetwornika umożliwia transmisję SPI z maksymalną częstotliwością 50 MHz, co pozwala na maksymalną częstotliwość odświerzania jego wyjść równą ok. 1,5 MHz. Transmisję przeprowadzimy z takim właśnie sygnałem zegarowym (jest to jednocześnie częstotliwość generatora kwarcowego dostępnego na płytce), ale częstotliwość odświeżania, a więc wysyłania kolejnych ramek, będzie równa 40 kHz. Kod źródłowy opisu bloku transmisji SPI jest na **list. 5.2**. Jest on oparty o rejestr przesuwny *sreg_out*, do którego po sygnale zezwolenia na zapis (*wr*) wpisywana jest ramka, a następnie jego zawartość jest przesuwana w lewo, przy czym ostatni bit tego rejestru jest połączony z linią *mosi*, a więc linią danych. Sygnał zegarowy uzyskany został poprzez bramkowanie zegara głównego z FPGA. Aby spełnić wymagania czasowe pomiędzy narastającym zboczem sygnału



Rys. 5.5 Oscyloskopowy obraz sinusoidy na wyjściu przetwornika C/A

sck a stanem ustalonym na linii *mosi*, faza zegara została odwrócona. Sygnał *busy* jest jednocześnie sygnałem *dac_cs* przetwornika.

Schemat układu generatora wraz z interfejsem SPI przedstawiono na **rys. 5.4**. Generator częstotliwości próbkującej 40 kHz (*sampling_generator*) jest podłączony zarówno do bloku rezonatora harmonicznego *sine_iir* (zezwalając na przeprowadzenie kolejnego kroku obliczeniowego) jak i modułu transmisji *spi* (w którym inicjalizuje transmisję). W listingu można odczytać konwersję z kodu uzupełnienia do 2 na kod naturalny dwójkowy (bez znaku) przez dodanie liczby 2048. Tuż za operacją dodawania następuje konstruowanie 32-bitowej ramki danych dla przetwornika cyfrowo-analogowego.

Na **rys. 5.5** przedstawia oscyloskopowy obraz sygnału na wyjściu przetwornika cyfrowo-analogowego

Projekt układu (zamieszczony na płycie CD), zrealizowany w systemie projektowym Xilinx ISE, działa na płytce uruchomieniowej *Spartan-3E Starter Kit HW-SPAR3E-SK-UK*.

Krzysztof Kasiński
krzysztof.kasinski@o2.pl
http://home.agh.edu.pl/kasinski

Literatura:
1. Texas Instruments TMS320C62x Algorithm: Sine Wave Generation. Application Report SPRA708 – November 2000.
2. Spartan-3E Starter Kit Board User Guide. UG230 (v1.0) 9.03.2006.
3 Joan Carletta, Robert Veillette, Frederick Krach, Zhengwei Fang, Determining Appropriate Precisions for Signals in Fixed-Point IIR Filters, 40th Design Automation Conference (DAC'03 pp.656), 2003