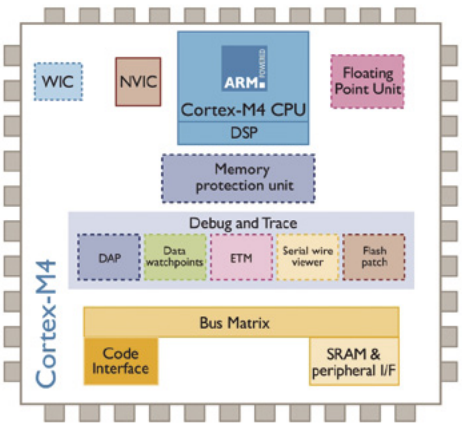
Siła rdzenia Cortex-M4F w nowych STM32F4

W poprzednich numerach EP zaprezentowano nowe mikrokontrolery z rodziny STM32F4 z rdzeniem Cortex-M4F. W niniejszym artykule skupimy się na tym, co stanowi o sile nowych układów – na rdzeniu Cortex-M4F.

Rodzina rdzeni Cortex jest podzielona na trzy segmenty. Grupa nosząca nazwę Cortex -A to rozbudowane i szybkie rdzenie z przeznaczeniem dla procesorów aplikacyjnych. Znajdują zastosowanie m. in. w elektronice konsumenckiej pokroju smartfonów, czy tabletów. Drugi segment stanowią rdzenie Cortex-R stworzone z myślą o aplikacjach czasu rzeczywistego, w których czasy wykonywania operacji muszą być deterministyczne. Z punktu widzenia „mikrokontrolerowca" najbardziej interesujący są przedstawiciele grupy Cortex-M. Należą do niej układy oznaczone cyframi od 0, 1, 3, 4.

Rdzenie M0 zostały stworzone z myślą o prostszych aplikacjach, w których do tej pory niepodzielnie królowały rozwiązania ośmiobitowe. Lepsza wydajność przy porównywalnej cenie z układami 8-bitowymi powinna Cortexom-M0 umożliwić stanięcie w szranki z mniejszymi braćmi. Jednak na chwilę obecną twarde ekonomiczne realia nieco utrudniają podjęcie walki o wartościowy kawałek tego fragmentu rynku. Często nie chodzi tu jednak o samą cenę mikrokontrolera (jeśli założymy, że są porównywalne), ale o koszty przeprojektowania aplikacji na nowe układy. W przypadku nowych urządzeń górę bierze przyzwyczajenie, doświadczenie i zwyczajna niechęć do podążania za nowymi technologiami.



Rys. 1. Schemat blokowy rdzenia Cortex-M4

Cortex-M1 to softrdzeń przeznaczony do implementacji w układach FPGA. Dotychczas największą popularność zdobył rdzeń sufiksem M3. Większość liczących się producentów MCU (oczywiście z wyjątkiem Mircochipa) ma w swojej ofercie mikrokontrolery z tym rdzeniem. Jednak nawet systemy głęboko wbudowane wymagają coraz większej mocy obliczeniowej. Dlatego firma ARM zaprojektowała mocniejszą wersję popularnych rdzeni M, jej głównym przeznaczeniem są aplikacje pracujące na granicy systemów wbudowanych i przetwarzania sygnałów. Używając ogólnie przyjętej nomenklatury, o układach wyposażonych w rdzenie Cortex -M4 można powiedzieć, że są to kontrolery sygnałowe (DSC – *Digital Signal Controller*).

Wsparcie dla DSP

Blokowy schemat rdzenia Cortex-M4 przedstawiono na **rysunku 1**. Jak widać blok FPU oraz kilka innych są opcjonalne. Od strony wsparcia dla przetwarzania sygnałów do dyspozycji są jednotaktowe instrukcje MAC (Multiply–accumulate). Długość danych poddawanych operacji MAC może mieć 16 lub 32 bity. Interesująca jest też możliwość wykonania instrukcji MAC na dwóch liczbach 16-bitowych jednocześnie. Wiadomo, że operacje matematyczne często wykonywane są na liczbach mniejszych, niż całe dostępne 32 bity. Wtedy to bardzo przydatne mogą okazać się instrukcje SIMD (Single Instruction – Multiple Data). Rdzeń Cortex-M4 umożliwia wykonanie instrukcji SIMD na danych 8 i 16-bitowych. Ponadto do dyspozycji jest również moduł sprzętowego dzielenia, który potrafi wykonywać dzielenie w ciągu 2 do 12 cykli.

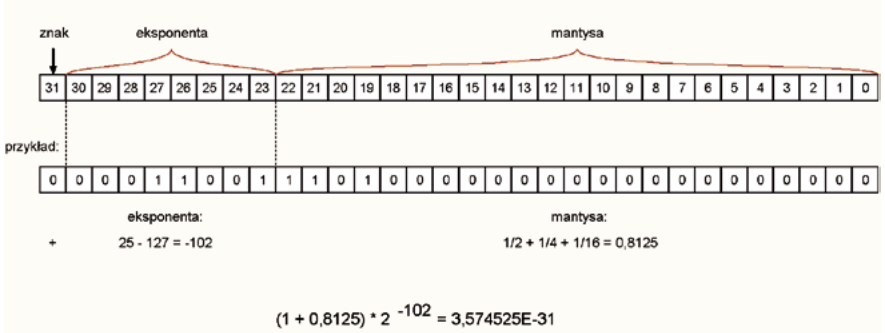
MPU

Z przedstawionego wcześniej rysunku 1 wynika, ze rdzenie Cortex-M4 mogą posiadać opcjonalnie jednostkę ochrony pamięci – MPU (Memory Protection Unit). Warto tutaj zaznaczyć, że mikrokontrolery STM32F4 posiadają taką jednostkę. Zadaniem MPU jest zarządzanie dostępem do chronionych obszarów pamięci. Takie zabezpieczenie ma przeciwdziałać przypadkowemu nadpisaniu krytycznego obszaru pamięci. Rdzenie Cortex-M4 umożliwiają utworzenie do ośmiu obszarów chronionych. Jednostka ochrony pamięci może być wykorzystywana przez system operacyjny czasu rzeczywistego (RTOS), a obszary chronione mogą być dynamicznie zarządzane przez RTOS.

Liczby zmiennoprzecinkowe

Programiści często używają pojęcia „liczba zmiennoprzecinkowa" (w Języku C są to typy float lub double) w odniesieniu do liczb posiadających części ułamkowe. Mają oczywiście co to tego pełną słuszność, ale liczba zmiennoprzecinkowa to twór dość skomplikowany i określenie „liczba z ułamkiem" jest daleko idącym uproszczeniem. Warto więc zrozumieć, czym owe typy „float" oraz „double" właściwie są.

Typ float jest nazywamy typem zmiennoprzecinkowym pojedynczej precyzji, jego reprezentacja to liczba 32-bitowa zgodna na przykład ze standardem IEEE 754. W tym standardzie reprezentacja typu float jest następująca:



Typ double jest określany jako liczba zmiennoprzecinkowa podwójnej precyzji. Charakteryzuje się długością 64 bitów, sposób kodowania i interpretacji jest taki sam jak dla liczb pojedynczej precyzji, zmianie ulegają długości pól eksponenty (wynosi 11 bitów) i mantysy (52 bity).

- arm_cortexM4lf_math.lib (Little endian and Floating Point Unit on Cortex-M4)
- arm_cortexM4bf_math.lib (Big endian and Floating Point Unit on Cortex-M4)
- arm_cortexM4l_math.lib (Little endian on Cortex-M4)
- arm_cortexM4b_math.lib (Big endian on Cortex-M4)
- arm_cortexM3l_math.lib (Little endian on Cortex-M3)
- arm_cortexM3b_math.lib (Big endian on Cortex-M3)
- arm_cortexM0l_math.lib (Little endian on Cortex-M0)
- arm_cortexM0b_math.lib (Big endian on Cortex-M3)

Rys. 2. Pliki *.lib dostępne w bibliotece CMSIS

FPU

Niezwykle istotnym wyposażeniem opcjonalnym omawianego rdzenia może być jednostka do obliczeń zmiennoprzecinkowych pojedynczej precyzji (FPU – Floating Point Unit). Rdzenie z FPU mają przypisaną nazwę Cortex-M4F, w ten typ wyposażone są na przykład mikrokontrolery STM32F4. Sprzętowe jednostki FPU to duży ukłon w kierunku konstruktorów zajmujących cyfrowym przetwarzaniem sygnałów o dużej dynamice. W takich przypadkach liczby zmiennoprzecinkowe bardzo się przydają, ponieważ odstęp pomiędzy dwoma liczbami jest zmienny. Dla małych wartości odstępy są również bardzo małe. Idąc w kierunku większych liczb odstępy ulegają zwiększaniu. Ujmując to dla porównania w przykład: załóżmy, że mamy dwie liczby stałoprzecinkowe 4 i 8. Pomiędzy tymi liczbami są możliwe tylko trzy wartości. W arytmetyce zmiennoprzecinkowej pojedynczej precyzji odstęp jest 10 milionów razy większy.

Moduł FPU z rdzenia Cortex-M4F potrafi wykonywać podstawowe operacje matematyczne (dodawanie, odejmowanie, mnożenie, dzielenie), wspomnianą już wyżej instrukcję MAC oraz pierwiastek kwadratowy. Możliwe są trzy tryby pracy jednostki obliczeń zmiennoprzecinkowych: tryb pełnej zgodności (Full-compliance mode), Flush-to-zero mode, tryb NaN (Default NaN). Pierwszy tryb zgodności oznacza, że FPU będzie pracować dokładnie tak, jak to opisuje standard IEEE 754. Ustawienie drugiego trybu będzie skutkowało tym, ze wszystkie tzw. liczby subnormalne podczas operacji będą traktowane jak zera. Pojęcie liczb „subnormalnych" nie jest oczywiste i na pewno wymaga kilku słów komentarza. Otóż do zbioru liczb subnormalnych zalicza się wszystkie liczby wokół zera, których nie da się przedstawić zapisem zmiennoprzecinkowym.

Ostatni tryb (Default NaN) określa zachowanie FPU w przypadku wystąpienia na wejściu lub wyjściu wartości NaN (Not a number). Jeśli jest aktywowany, to w większości przypadków operacji, gdzie zaangażowana jest wartość NaN zostanie zwrócona również wartość NaN.

CMSIS dla DSP

CMSIS (Cortex Microcontroller Software Interface Standard) jest biblioteką niezależną od producenta układów, zawiera duże wsparcie dla operacji specyficznych dla rdzeni Cortex. Ponieważ w kręgu naszych zainteresowań leżą operacje DSP, to skupimy się wsparciu właśnie dla nich.

Część biblioteki zawierająca funkcje dla DSP jest dostarczana oddzielnie dla rdzeni M0, M3 oraz M4. Oczywiście największą wydajność przetwarzania osiągnie się z użyciem rdzenia Cortex-M4F, który to większość czasochłonnych obliczeń jest w stanie wykonywać sprzętowo. Pozostałe dwa rdzenie muszą specyficzne operacje przetwarzania sygnałów emulować programowo.

Wszystkie pliki .lib można znaleźć na **rysunku 2**. Dodatkowo dostępne są źródła, jeśli zaistniałaby potrzeba przekompilowania bibliotek DSP. Zestaw funkcji obejmuje takie obszary jak: podstawowe i szybkie funkcje matematyczne, funkcje dla liczb zespolonych, filtry, funkcje operujące na macierzach, transformaty, funkcje wspomagające projektowanie aplikacji z silnikami elektrycznymi, funkcje statystyczne i interpolacji. Ważne jest również to, że biblioteka została podzielona do działań na liczbach 8-bitowych, 16-bitowych, 32-bitowych oraz 32-bitowych zmiennoprzecinkowych. Dzięki temu lepiej można dopasować wywoływane fragmenty kodu do specyfiki aplikacji.

Funkcje biblioteczne zostały zadeklarowane w pliku arm_math.h, a zatem aby z nich skorzystać należy dołączyć ten plik do swojego projektu oraz dolinkować stosowne pliki bibliotek. Wyboru rdzenia dokonuje się poprzez definicje preprocesora, dla Cortex-M4 będzie to MACRO_ARM_MATH_CM4, dla pozostałych rdzenie będzie to analogicznie MACRO_ARM_MATH_CM3 i MACRO_ARM_MATH_CM0. Jak widzimy, powyższe definicje nie określają, czy rdzeń posiada jednostkę zmiennoprzecinkową. W celu zaznaczenia, ze używany mikrokontroler posiada rdzeń z FPU (czyli Cortex-M4F) należy makro __FPU_PRESENT zainicjalizować wartością 1.

Oddzielnie tylko dla rdzenia Cortex-M4 stworzone zostały funkcje używające instrukcji SIMD. Takich funkcji jest w sumie aż 59. Aby pokazać możliwości instrukcji SIMD dla przykładu omówimy jedną z nich: SMLALD, która oprócz jednoczesnego wykonania mnożenia na dwóch 16-bitowych liczbach dodaje wyniki mnożenia i trzecią liczbę do siebie i generuje 64-bitowy wynik. Jest to zatem wariacja instrukcji MAC i SIMD. Sposób użycia SMLALD z pewnością wyjaśni poniższa linia kodu:

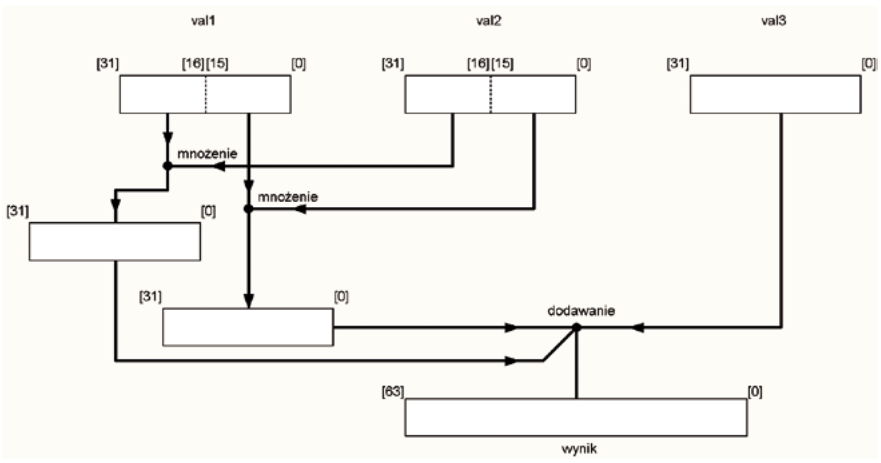
```
uint64_t __SMLALD(uint32_t val1,
uint32_t val2, uint64_t val3);
```

Parametry val1 i val2 to odpowiednio mnożne i mnożniki. Są to liczby 32-bitowe, a więc młodsze dwa bajty i starsze dwa bajty są traktowane jako różne liczby. Trzeci parametr val3 jest dodawany na samym końcu. Całą operację SMLALD przedstawia **rysunek 3**. Istotną informacją jest, że przepełnienie może wystąpić tylko na etapie dodawania wartości 64-bitowych, a nie podczas mnożenia. Fakt przepełnienia nie jest odnotowywany. Powyżej przedstawiono tylko jedną funkcję wykorzystującą SIMD, a pozostało jeszcze 58, stąd łatwo zauważyć, ze twórcy CMSIS wykonali sporo pracy przygotowując bibliotekę DSP.

Przykłady w CMSIS

Oprócz samych bibliotek i ich kodów źródłowych, do biblioteki CMSIS dołączonych jest kilka przykładowych projektów z tematyki DSP. Obok standardowych przykładów takich jak transformata FFT, czy realizacja filtru FIR, mamy do wglądu kod aplikacji pięciopunktowego equalizera. Wszystkie przykłady pozbawione są niestety wszelkiego interfejsu, czyli innymi słowy obliczenia wykonywane są na tablicach wartości zapisanych w pamięci. Dlatego jedynym wynikiem wykonywania programów jest ocena zgodności, czy otrzymane wartości sygnałów są zgodne ze wzorcami. Nie jest to niestety zbyt efektowne. Dlatego, jeśli będzie duże zainteresowanie rzeczywistymi aplikacjami DSP dla mikrokontrolerów z rdzeniem Cortex-M4F (konkretniej chodzi o układy STM32F4), to zostanie przygotowanych kilka projektów pokazujących „w akcji" działanie bibliotek CMSIS dla DSP.

Krzysztof Paprocki, EP



Rys. 3. Operacja SMLALD