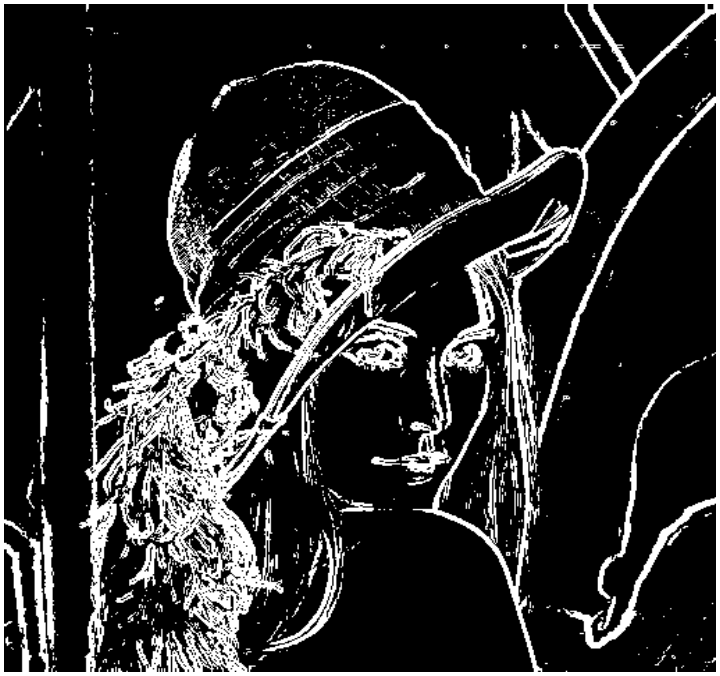
Instrukcja do zajęć laboratoryjnych

Budowa i programowanie DSP, kierunek: elektronika, II stopień

Autor: dr inż. Piotr Zając



Uwaga: niniejsza instrukcja zakłada, że

- studenci posiadają na swoich kontach odpowiednio skonfigurowaną maszynę wirtualną ze środowiskiem DVSDK firmy Texas Instruments,
- studenci potrafią połączyć płytkę Devkit8500 z komputerem,

oraz posiadają wiedzę z następujących zagadnień:

- uruchomienie i programowanie płytki Devkit8500,
- kompilacja skrośna z wykorzystaniem środowiska DVSDK firmy Texas Instruments,
- podstawowe informacje o filtrach cyfrowych i przetwarzaniu obrazów.

Zadanie 1 (4 godz.)

Zadania wstępne:

- Ze strony podanej przez Prowadzącego zajęcia należy ściągnąć przykładowy projekt i przeanalizować sposób jego kompilacji (pliki Makefile oraz linker.cmd).
- Zwrócić uwagę na fragment programu wykorzystujący do obliczeń rdzeń DSP. Przeanalizować różnice między obecnym projektem i, a projektem wykorzystującym do obliczeń tylko rdzeń ARM (poznanym na poprzednich zajęciach). W szczególności należy zwrócić uwagę na:
    - funkcję c6accel_init()
    - alokację specjalnych buforów w pamięci współdzielonej (pSrcBuf_16bpp i inne)
- Zapoznać się z dokumentami „C6Accel whitepaper” oraz „C6Accel User Documentation”, a następnie przeanalizować i zrozumieć w jaki sposób rdzeń ARM ma możliwość uruchomienia funkcji na rdzeniu DSP.
- Na przykładzie funkcji C6accel_IMG_Sobel_3x3_8 prześledzić cały ten proces, czyli kolejność wykonywanie poszczególnych funkcji (od uruchomienia funkcji Sobel na rdzeniu ARM do uruchomienia właściwych obliczeń na rdzeniu DSP. Wskazówka: kod biblioteki C6accel znajduje się w folderze /home/dmcs/dvsdk/c6accel_1_01_00_07

Zadanie:

- Zrealizować wykrywanie krawędzi na strumieniu wideo za pomocą filtru Sobela, wykorzystując do obliczeń rdzeń DSP.
- Można wzorować się na kodzie programu w folderze /home/dmcs/dvsdk/dvsdk-demos_4_02_00_01/omap3530/edge_detection
- Zapoznać się z C6Accel API Reference Guide i użyć kolejno funkcji:
    - C6accel_IMG_yc_demux_le8_8 (do rozdzielenia obrazu w formacie YUYV na trzy oddzielne tablice Y, U ,V)
    - C6accel_IMG_sobel_3x3_8 (na tablicy Y)

- C6accel_IMG_ycbcr422sp_to_ycbcr422ile (do złożenia obrazu w formacie YUYV z dwóch oddzielnych tablic Y i UV)

- Wskazówka 1: bufor z chrominancjami dla obrazu wyjściowego należy ustawić samodzielnie tak, aby obraz wyjściowy był w skali szarości
- Wskazówka 2: należy pamiętać o tym, że każda funkcja wykorzystująca DSP musi wykorzystywać bufory w pamięci współdzielonej
- Wskazówka 3: w ostatniej funkcji składającej luminancję i chrominancję w jeden obraz wyjściowy kolejność argumentów jest błędnie podana w C6Accel API Reference Guide

Zadanie dodatkowe:

- Zrealizować filtr Laplace'a na strumieniu wideo za pomocą funkcji splotu i samodzielnie zdefiniowanej maski. Uwaga: należy użyć funkcji C6accel_IMG_conv_3x3_i8_c8s_Frame(), która operuje na całym obrazie w przeciwieństwie do funkcji C6accel_IMG_conv_3x3_i8_c8s(), która operuje na jednej linii obrazu.

Zadanie 2 (6 godz.)

Zadanie:

- Napisać własną funkcję wykonywaną na rdzeniu DSP i użyć jej na procesorze ARM. Funkcja my_filter() powinna dodawać do luminancji obrazu wejściowego wartość podaną jako argument i generować odpowiednio zmieniony obraz wyjściowy. Wskazówka: należy wzorować się na funkcji pz_function() napisanej przez Prowadzącego.
- W celu wyszukania wszystkich miejsc w których występuje kod związany z funkcją pz_function() można wykorzystać komendę grep (polecenie `grep -r „pz_function" .` lub `grep -r „PZ_FUNCTION" .`). Dokładnie w tych samych miejscach należy dodać odpowiedni kod dla własnej funkcji my_filter(). Uwaga: należy pamiętać o tym, żeby dla własnej wygody stosować się do tej samej konwencji nazewnictwa funkcji i zmiennych, która występuje w kodzie.
- Po dodaniu kodu należy pamiętać, żeby przed uruchomieniem programu, oprócz własnego projektu skompilować również bibliotekę c6accel w folderze /home/dmcs/dvsdk/c6accel_1_01_00_07 i skopiować wynikowy plik /home/dmcs/dvsdk/c6accel_1_01_00_07/soc/packages/ti/c6accel_unitservers/omap3530/c6accel_omap3530.x64P na płytkę Devkit.

Zadanie 3 (2 godz.)

Zadanie:

- Na podstawie informacji podanych w dokumencie „TMS320C6000 Compiler User Guide” (od rozdziału 7.5.5) zapoznać instrukcjami umożliwiającymi wykonywanie operacji typu SIMD na rdzeniu DSP (tzw. „compiler instrinsics”). Przeanalizować i zrozumieć użycie tych instrukcji na przykładzie funkcji complextorealimg().
- Przepisać program z poprzedniego zadania z wykorzystaniem instrukcji „compiler instrinsics” tak, żeby wykonywał on się szybciej. Wskazówka: należy napisać główną pętlę tak, żeby dodawać stałą wartość do luminancji **dla czterech pikseli jednocześnie**.

Zadanie 4 (6 godz.)

Zadanie polega na zrealizowaniu na procesorze DSP dyskretnej transformaty Fouriera sygnału jednowymiarowego z użyciem algorytmu FFT (Fast Fourier Transform). W szczególności należy wykonać następujące kroki:

- Po stronie procesora ARM wypełnić tablicę N=256 próbkami sygnału prostokątnego o okresie równym 64 próbkom. Przekazać tablicę jako argument wejściowy do poprzednio napisanej funkcji „my_filter”. Dalsze operacje powinny być wykonywane już na procesorze DSP.
- Ze względu na wykorzystywane funkcje do obliczenia FFT, należy przekopiować dane z tablicy otrzymanej jako argument funkcji „my filter” do tablicy liczb typu „short”. Drugim wymogiem jest zapewnienie, że liczby w tablicy są zespolone, przy czym najpierw w tablicy występuje część rzeczywista liczby a na następnym miejscu tablicy występuje cześć urojona (która będzie zawsze zerowa). Uwaga: tablica powinna mieć więc w sumie 2*N elementów typu „short”. Amplituda sygnału zostanie podana przez Prowadzącego zajęcia.
- Zapoznać się z funkcją gen_twiddle_fft16x16(), która generuje tablicę ze współczynnikami pomocniczymi dla obliczeń FFT. Użyć jej do odpowiedniego wygenerowania współczynników.
- Zapoznać się z działaniem funkcji DSP_fft16x16() na podstawie dokumentacji podanej przez Prowadzącego zajęcia. Użyć funkcji do obliczenia FFT sygnału prostokątnego, otrzymany wynik przekazać jako argument wyjściowy funkcji „my_filter” (uwaga: należy pomyśleć w jaki sposób przekazać liczby typu „short” w tablicy liczb typu „char”).
- Wyświetlić wyniki (po stronie procesora ARM) i przeanalizować czy są one zgodne z oczekiwaniami.

Jeżeli powyższe punkty udało się zrealizować pomyślnie, należy wykonać następujące punkty (należy je wykonać rozdzielnie):

- obliczyć moduł dla każdej liczb zespolonej otrzymanej w wyniku FFT, obliczony moduł przekazać argument wyjściowy funkcji „my_filter” i wyświetlić po stronie procesora ARM
- po wykonaniu FFT, na otrzymanych danych zrealizować odwrotną transformatę Fouriera z użyciem algorytmu IFFT (funkcja DSP_ifft16x16()). Sprawdzić, czy otrzymany wynik jest zbieżny z teorią (tablica z wynikami powinna być taka sama jak pierwotna tablica z próbkami czasowymi). Wytłumaczyć skąd mogą brać się różnice między danymi w tablicach. Uwaga: w celu wykonania operacji IFFT potrzebna będzie funkcja gen_twiddle_ifft16x16(), zostanie ona podana przez Prowadzącego zajęcia, należy dodać ją do pliku gen_twiddle.c.