Heterogén SoC rendszerek

Házifeladat

2025.

Tartalom

1 Fejlesztői/futtató környezet

A házi feladat 1 – 3 részfeladatai elkészíthetőek akár AMD64(x86), akár ARM64 platformon.

- AMD64 esetében a fejlesztés a gyakorlatokon is használt környezetben történik. Az elkészített alkalmazások vagy az IE321/IE226 laborban, vagy pedig a saját PC-teken tesztelhető natív Linux vagy Windows Subsystem for Linux használatával.
- Aki rendelkezik valamilyen saját, Linux alapú ARM64 számítógéppel, az használhatja azt a fejlesztésre, ebben az esetben a folyamat megegyezik a gyakorlatokon megismerttel. Alternatív lehetőség a Termux használata Android-on – ebben az esetben mind a fordítás, mind pedig a tesztelés Termux-on történi.

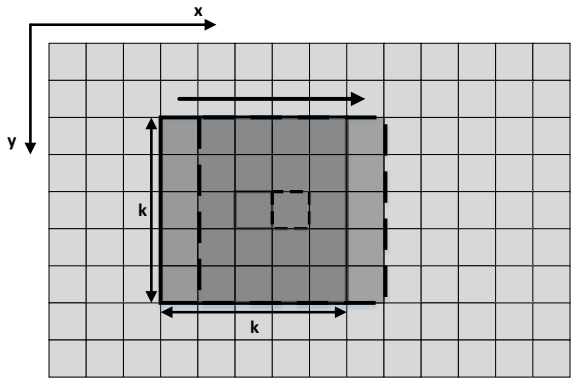
VSCode, WSL2, Termux és a szükséges library-k ügyében a megfelelő PDF-ben található információ (Teams csoport).

A 4. részfeladathoz az előző félévben is használt AMD Vitis fejlesztői környezetre lesz szükség.

2 A megvalósítandó feladat: 5x5 medián szűrő

A házifeladat során egy kétdimenziós 5x5 méretű medián szűrőt kell megvalósítani különböző platformokon és technológiákkal. A szűrő bemenete RGB komponenseket tartalmazó kép, illetve videó folyam.

A medián szűrő egy nem lineáris szűrő, amely a szűrő kernel által lefedett értékeket nagyság szerint sorba rendezi, s kimenetre a rendezett értékek közül a középső kerül. A teljes kép szűréséhez a szűrőablakot a teljes képen „végigcsúsztatjuk”.



Bár általános képfeldolgozás során általában a színkomponensekből számított világosság komponens alapján történik a rendezés, **az egyszerűség kedvéért a házi feladatban minden színkomponensre külön-külön elvégezzük a szűrést**. A kép szélein a hiányzó bemeneteket fekete pixelekkel helyettesítjük.

Mivel rendkívül számításigényes feladat, így a medián szűrésre számos különböző algoritmus létezik. A házifeladat megvalósítása során jól párhuzamosítható és vektorizálható algoritmusra van szükségünk, a javasolt megoldások:

- Batcher Odd-Even Merge Sort:
  https://en.wikipedia.org/wiki/Batcher_odd%E2%80%93even_mergesort
- AltiVec 5x5 optimalizált rendezőhálózat: a cikket lásd Teams-en.

A két megoldás közül az alapvető optimalizációkat alkalmazva az Altivec némileg gyorsabb, de további lépésekkel a Batcher is gyorsabbá tehető. Megvalósítás szempontjából a Batcher az egyszerűbb. **Minden részfeladatban ugyanazt a választott algoritmust kell megvalósítani!**

3 Feladatok

3.1 Skalár referencia implementáció (1p)

Első lépésben készítse el az algoritmus skalár, sztenderd C/C++ implementációját, azaz sztenderd C/C++ típusok és függvények használatával írja meg a kódot. Ezen lépés célja, hogy legyen egy referencia implementáció, amely megkönnyíti a további verziók esetleges lépésenkénti debug-olását.

A PDF dokumentáció tartalmazza:

- A megvalósított algoritmus leírását ábrákkal és szövegesen
- A median_filter_scalar() függvény olvashatóan tördelt, színezett, kommentezett forráskódját

Beadási határidő: külön nincs, a 3.2, 3.3, 3.4 részfeladatok közül az első beadásának időpontjában kell feltölteni.

3.2 Vektorizált, többszálú implementáció (8p)

Készítse el az előbbi implementáció lehető legoptimálisabb többszálú, vektorizált megoldását. A vektorizáció során AMD64 platform esetén legfeljebb AVX2 (és régebbi) utasítások használhatók. Megengedett mind intrinsic függvények, mind pedig assembly használata.

A PDF dokumentáció tartalmazza:

- A vektorizáció megvalósításának elvét ábrákkal és szövegesen
- A median_filter_vector() függvény olvashatóan tördelt, színezett, kommentezett forráskódját
- A referencia és a saját megvalósítás futási idő eredményét képernyő mentés formájában

Beadási határidő: kb. a kapcsolódó gyakorlat után 4 héttel, pontosan lásd MIT HF portál (hf.mit.bme.hu)

3.3 GPU OpenCL implementáció (8p)

Készítse el a medián szűrőt megvalósító, GPU-ra optimalizált OpenCL kernel-t.

A PDF dokumentáció tartalmazza:

- A kernel függvény olvashatóan tördelt, színezett, kommentezett forráskódját
- A hatékonysággal kapcsolatos magyarázatokat szövegesen és ábrákkal (Work-group méret, Global Memory hozzáférés, illetve amennyiben releváns Local Memory méret és Local Memory hozzáférési mintázat)
- A futtatáshoz használt GPU típusát
- A referencia és a saját megvalósítás futási idő eredményét képernyő mentés formájában

Beadási határidő: kb. a kapcsolódó gyakorlat után 4 héttel, pontosan lásd MIT HF portál (hf.mit.bme.hu)

3.4 Vitis HLS C implementáció (8p)

Készítse el a medián szűrő teljesen párhuzamos implementációját (minden órajelben képes egy kimeneti pixel kiszámítására) Vitis HLS-ben. A szintézisre szánt C++ függvény működését ellenőrizze egy C++ testbench segítségével.

Helyes működés esetén készítsen a Vitis HLS segítségével IP-t, s implementálja azt a Teams-ről letölthető HDMI loopback projektben, majd tesztelje a működést a Logsys Kintex-7 kártyán.

A megvalósítás során ügyeljen rá, hogy míg az eddigiekben a teljes bemenet rendelkezésre állt a memóriában, addig az FPGA implementáció esetében a számításhoz szükséges adatok eltárolását is a C kódnak kell megvalósítania.

A dokumentáció tartalmazza:

- A megvalósított hardver blokkvázlatát
- A C szimulációhoz használt függvény olvashatóan tördelt, színezett, kommentezett forráskódját
- A hardver szintézisre szánt függvény olvashatóan tördelt, színezett, kommentezett forráskódját
- A használt pragma-k magyarázatát
- A C szintézis időzítési és erőforrás eredményét
- A teljes terv place&route implementációs eredményeket (időzítés, erőforrás igény – LUT, BRAM, DSP).

Beadási határidő: a szorgalmi időszak vége.

4 Beadandó fájlok

- Minden feladathoz egyetlen ZIP fájlt kell feltölteni a BME MIT HF portálra (hf.mit.bme.hu).
- A 3.1 – 3.3 feladatokhoz ebben az alábbiaknak kell szerepelnie egy VEZETEKNEV_KERESZTNEV könyvtárban:
    - A feladat rövid dokumentációja, lásd fentebb.
    - Egyetlen futtatható állomány, amely a megvalósított feladatot tartalmazza
        - A bemeneti kép kihagyható a HF portál méret korlátja miatt.
    - Egy src könyvtárban a teljes forráskód, amiből a projekt fordítható.
- 3.4 feladat esetében a ZIP fájl tartalma:
    - A felesleges fájloktól megszabadított Vivado projekt
    - A felesleges fájloktól megszabadított Vitis HLS projekt
    - A fentebb részletezett dokumentáció
    - Ezt a részfeladatot a félév végén élőben is be kell mutatni!

5 Értékelés

A megszerzett házi feladat pontszám hozzáadódik a vizsga eredményéhez, de legfeljebb 100 pont érhető el.

A helyes megoldás azt jelenti, hogy a generált kép pixelről pixelre megegyezik a referencia képpel, nem pedig azt, hogy „ránézésre ugyanolyan”!

5.1 Pontozás

A 2.2 és 2.3 feladatok pontozása – helyes kimeneti eredmény esetén – a beadott megoldás referencia megoldáshoz képest mért teljesítményén múlik.

- Helytelen kimenet: 0 pont
- <57%: 1 pont
- 57% - 64%: 2 pont
- 65% - 70%: 3 pont
- 71% - 78%: 4 pont
- 79% - 84%: 5 pont
- 85% - 92%: 6 pont
- 93% -99%: 7 pont
- >=100%: 8 pont

A kiértékeléshez használt hardver:

- AMD64 megoldás esetén:
  - A 2.1 és 2.2 feladatok esetén egy Intel Core i5 8600K processzoron végezzük a méréseket DDR4@3200 memória társaságában.
  - A 2.3 esetén egy NVIDIA Titan Xp GPU-n és egy Intel UHD Graphics 630 GPU-n futtatjuk a beadott megoldást, az arányaiban jobb eredményt vesszük figyelembe.
- ARM64 megoldás esetén:
  - A gyakorlaton is használt Orange Pi 5B.

A 2.4 feladat esetén:

- 2 pont, ha a C szimuláció eredménye helyes
- 3 pont, ha ezen túlmenően a C szintézis időzítési paraméterei megfelelőek
- A további pontok a megoldás minőségétől függenek

Plágium gyanúja esetén a HF eredménye 0 pont.

5.2 Megajánlott jegy

Amennyiben valaki minden részfeladatot megfelelő minőségben elkészít, úgy megajánlott jegyet kap.

- A megajánlott osztályzat Jó (4) amennyiben a HF összpontszáma legalább 14.
- A megajánlott osztályzat Jeles (5) amennyiben a HF összpontszáma legalább 18.