

Elektrosztatikus mérés szimulációja multiprocesszoros környezetben

Bakró-Nagy István

Szélessávú Hírközlés és Villamosságtan Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem
Budapest
bakro.istvan@gmail.com

Reichardt András

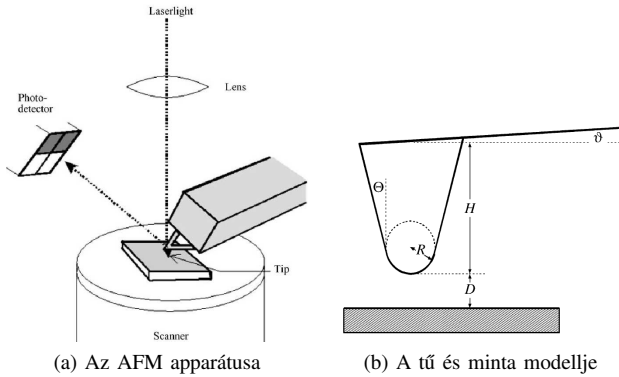
Szélessávú Hírközlés és Villamosságtan Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem
Budapest
reich@evt.bme.hu

Kivonat—Fémezett, töltött felület felett lévő fémtűre ható erő számítását végeztük el. A szimulációk során a lehetséges párhuzamosításokat felhasználva a számítási időt csökkentettük. Kidolgoztunk egy egyszerű szimulációs keretrendszert a felületmérés pontatlanságainak kiegyensúlyozására, az elérhető felbontás javítására.

Index Terms—numerikus szimuláció, parallel computing, OpenCL, AFM

I. BEVEZETÉS

1986-ban Binning demonstrálta az atomerő mikroszkóp (AFM) ötletét [1], ami mára a nanotechnológia egyik legfontosabb eszköze lett. Felhasználható képalkotásra, nanolitográfiára és adott anyag alakítására [2]. Az AFM apparátusa az adott minta felülete és a felette pásztázó kantilever végére erősített tű kölcsönhatásának vizsgálatát végzi. (lásd 1b. ábra) A felület és a tű közötti domináns kölcsönhatás határozza meg, hogy az anyag melyik fizikai mennyiségét kaphatjuk meg.



1. ábra. Az AFM apparátusa látható az (a) ábrán. A lázernyaláb a kantilever felületéről tükröződve egy fotódetektorra irányul. A tű pozícióját ez alapján nagy pontossággal ismerjük. A (b) ábrán a minta és a felette lévő tű modellje látható a kapacitás analitikus számításához. A tű R sugarú H magasságú és D távolságra van a mintától. (Forrás: [3])

Az AFM felhasználása kontakt illetve kopogtató üzemmódú lehet. A kontakt mód során a felületen végighúzzuk a tűt és mérjük a z irányú elmozdulását. Így képesek vagyunk a minta felületén lévő atomok elrendezéséről magasságtérkép adni. Kopogtató mód [4] során a tűt elemeljük a mintától és f frekvenciával rezgetjük. A letapogatás során az átlagos minta-tű távolságot a kontakt módú magasságtérkép felhasználásával konstans értéken tartjuk. A kantilever dinamikáját ismerve a rezgetetés frekvenciájának eltéréséből számítható a tűre ható erő. Ezen erő nagyságát két tényező befolyásolja:

- a) a minta és a tű közötti feszültség,
- b) a minta felületi töltéssűrűség eloszlása.

A cikkben a felületi töltéssűrűség mérését tekintjük célunk. A [3] szerint az erő a) komponensét a minta és a tű közötti kapacitásból a (1) szerint származtathatjuk.

$$F_s = -\frac{dE}{dD} = -\frac{d(CV^2/2)}{dD} = -\frac{1}{2} \frac{dC}{dD} V^2 \quad (1)$$

Ha a minta pásztázása során ezen a) erőkomponens konstansnak mondható, tehát a felületi érdesség kicsi, akkor a töltéssűrűg mérésében állandó hibát okozva eliminálható. Az (1) számításában a kritikus elem a kapacitás értéke, amit numerikus számítás mellőzése esetén a [5] szerinti analitikus eredményt használhatjuk fel. A tű formáját a (1b ábra) szerintinek veszi és a mintát sík felületének feltételezi. Ezen utóbbi feltételezés legtöbb esetben helyénvaló, viszont a minta nagyfokú érdessége és a tű egyedi formája esetén érvényét veszti. Ilyen esetben a kapacitás értéke mintáról mintára változik és állandó hiba helyett, a mérést zajként terheli. A kapacitás numerikus szimulációjával ezen zajt is ki lehet küszöbölni. Persze ezen szimulációt minden egyes mérési pontban el kell végezni, aminek a kivitelezése csak multiprocesszoros környezetben lehetséges elfogadható idő alatt.

II. A FELADAT

A minta egy fémezett felület, aminek magasságtérképét mérések eredményeként ismerjük adott pontossággal. A mérések egy négyzetes háló felett történtek, amelynek a háló mindkét irányban $\delta_x = \delta_y = 120nm$ azonos felbontása, illetve a magasság $\delta_z = 20nm$ felbontása volt. A töltéssűrűség méréséhez szükséges második pásztázás során a tűt felemeljük és a mintához képest V_{tu} potenciálra kapcsoljuk. Ezután a fémezett felületről mindig azonos távolságra tartjuk és ezen hosszú hegyes tűre ható erőt mérjük.

Végző cél olyan szimulátor építése, amelynek segítségével közel valós időben lehetséges felületmérés alapján a felületi töltéseloszlásról korrigált/pontosabb információt kapni.

A cikkben felhasznált mérési eredmény (2. ábra) egy 512×512 méretű szürkeárnyalatos *.tiff állomány, amely értékei 0 – 255-ig terjed.

III. SZIMULÁCIÓ

III-A. Fizikai probléma matematikai formalizálása

A megoldandó feladat egy elektrosztatikus feladat. A minta és a tű közötti térben nincsenek töltések, így itt a Poisson egyenlet helyett a Laplace-egyenlet (2) érvényes.

$$\Delta V(x, y, z) = 0 \quad (2)$$

Az egyenletet későbbiekben részletezett (III-B) megfontolások végett egy redukált 3D-s téren oldjuk meg. Ezen 3D-s térré egy inhomogén ponttérrel illesztünk, amelynek vízszintesen $d_x = d_y$, függőlegesen



2. ábra. Méréssel kapott felület (balra) és szimulációval kapott töltéstérkép

d_z a felbontása. A függőleges felbontás megegyezik a használt AFM apparátus felbontásával $d_z = \delta_z$. Az így kapott térbeli háló minden pontjához hozzárendeljük az $V_{i,j,k} \simeq V(id_x, jd_y, kd_z)$ potenciált. Dirichlet határfeltételek a felület fémezése, amely zérus potenciálú és az adott (V_{tu}) potenciálú tű fémes felülete. A térnek a minta felületétől különböző határfelületén homogén Neumann feltételt alkalmazunk a szimmetriák (végtelen tér) és a töltésmentesség miatt.

Az így adódó lineáris egyenletrendszer megoldására lehetséges direkt és iteratív megoldó algoritmusokat alkalmazni. A párhuzamosítási szándékok miatt az iteratív megoldást választottuk, mivel a multiprocesszoros környezetek tipikusan kevés fajlagos-memóriával¹ rendelkeznek. Ekkor nem teljesen pontos megoldást kapunk, azonban gyorsabban juthatunk el a kívánt eredményhez. A számítási pontosság növelhető az iterációt leállító konvergencia követelmény keményebb megszabásával, ami persze több iterációt jelent.

Az iteratív megoldás során a megoldás aktuális értékének kiszámításához az előző megoldásból indulunk ki. A (2) egyenletben szereplő deriválást az elsőrendű Taylor közelítés alkalmazásával a (3) 6-pontos sémát kapjuk.

$$V_{ijk}^{n+1} = \Delta_1 \cdot (V_{i-1,j,k}^n + V_{i+1,j,k}^n + V_{i,j-1,k}^n + V_{i,j+1,k}^n) + \Delta_2 \cdot (V_{i,j,k-1}^n + V_{i,j,k+1}^n) \quad (3)$$

ahol $V_{i,j,k}^n$ az az n -dik iterációs lépésben az i, j, k indexű pontban mérhető potenciált jelöli, Δ_1 a vízszintes felbontásból, Δ_2 a függőleges felbontásból adódó állandó.

III-B. Szimuláció felépítése

A felületmérés során a vízszintes felbontás jóval kisebb, mint a függőleges felbontás $d_x = d_y \gg d_z$. A Coulomb-kölcsönhatás a távolság négyzetével fordítottan arányos, így az előbb említettek értelmében egy mérési pont szomszédjait, pontosabban egy redukált környezetét szükséges csupán szimulálni. (Másképpen megfogalmazva a vízszintes mérési pontok távolsága jóval nagyobb mint a Coulomb-kölcsönhatás effektív távolsága.) Ezzel az elhanyagolással már numerikusan kezelhető bonyolultságú problémára jutunk.

Ezen módon egy mért pont 3×3 -as környezetét vesszük figyelembe, a középső pont felett lévő elektródát (tűt) feltételezve. A szimulációs tér alsó felületét a mért magassági 3×3 mintákból kell meghatároznunk. Mivel ezen pontokra úgy is tekinthetünk, mint a minta magasság-függvényének a mintavételezésével kapott mintáira, így a küzbünső pontokat interpolációval/ideális aluláteresztéssel kapjuk meg. Az interpoláció N_{ip} faktorával² lehetséges a szimulációs tér vízszintes felbontását $d_x = d_y = \delta_x/N_{ip}$ megkapni.

¹Fajlagos alatt az egy szimulációra jutó memóriát értem. (Természetesen ezen szimulációk egyszerre futnak, így a fajlagos-memória akumulálódik.)

² N_{ip} -szeresére növeljük a pontok számát.

A szimulációk során a felület magasságának mérési adatait már ismertnek feltételezzük. A teljes magasságtérkép pontjait külön-külön vizsgáljuk. Egyetlen pontban a mérési eredmény kiszámításának lépései az alábbiak :

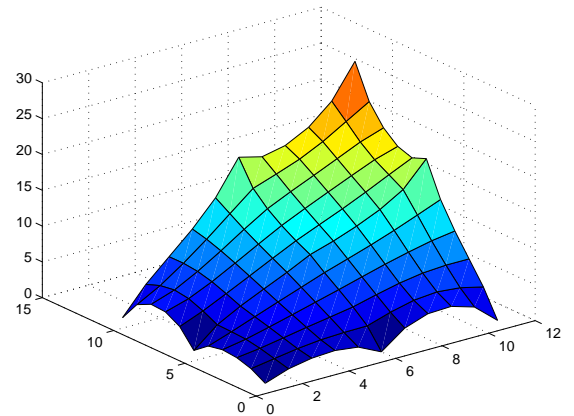
- 1) A pont körüli felület 3×3 -as mérési részének megállapítása,
- 2) Közbeső (virtuális) mérési pontokkal a belső felbontás növelése interpolációval,
- 3) Szimulálandó tér méretének számítása,
- 4) Direkt/iteratív megoldó algoritmussal a tér meghatározása, a tűre ható erő számítása illetve a tű alatti töltésmennyiség számítása,
- 5) Adatok mentése.

IV. A SZIMULÁTOR BEMUTATÁSA

A prototípus algoritmus fejlesztése MATLAB környezetben történt, ami később referenciaként szolgál. Alap MATLAB utasításokat használva több órát vesz igénybe a szimuláció futtatása. A MATLAB Parallel Toolbox-nak segítségével a szimulációt lehetséges párhuzamosan több processzormagon futtatni. Ezzel párszoros sebesség növekedés érhető el. A következőkben magát az algoritmust és az OpenCL keretrendszerben történő implementációját mutatjuk be. Majd az eredmények bemutatása során kerül összevetésre kerül a MATLAB referencia, a MATLAB Parallel Toolbox segítségével, az OpenCL processzoron és az OpenCL GPU-n való futtatási ideje.

IV-A. A lépések részletezése

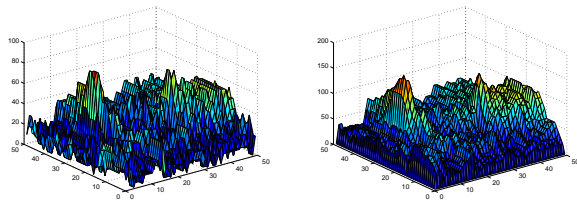
IV-A1. Interpoláció: A korábban elmondottak alapján a felületet további mérési pontokkal egészítjük ki. Az extra mérési pontokat a legegyszerűbb síklapos közelítéssel alkothatjuk meg, a magasságmérési felbontás figyelembevételével. A szimulátorban egy általánosabb módszert alkalmazunk, ami egy 2D-s mozgó átlagoló szűrővel való simítás. A szűrővel aluláteresztést tudunk elérni, ami a minta magasságának mintavételezése utáni rekonstrukcióját jelenti. Egy ilyen interpoláció eredményét láthatjuk a 3. ábrán.



3. ábra. 3×3 mérési pont 11×11 pontba való interpolációja

IV-A2. A szimulálandó tér mérete: A szimulálandó tér (hasáb) magasságát a következő két mennyiség közül a nagyobbikkal határoztuk meg:

- Középső pont fölött lévő tű közepének magassága,
- A (3×3) környezet legalacsonyabb és legmagasabb pontjának különbsége.



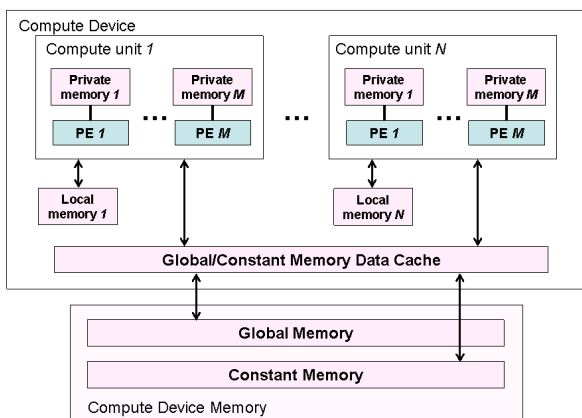
4. ábra. A magasságmérés eredményének részlete (balra) és a hozzá tartozó szimulálandó tér magassága.

IV-A3. Iteratív megoldó algoritmus: Az iterációhoz a térháló pontjaihoz két mátrixot (tömböt) rendelünk, ami a pontok potenciáljának aktuális (U_{now}) és előző (U_{prev}) értékeit tartalmazza. Az aktuális értékeket (??) szerint számítjuk, majd az egész térre számítjuk az előzővel vett különbségének négyzetösszegét (normáját). E mérték képviseli a konvergencia szintjét, amit az iteráció során vizsgálva jutunk el a kívánt konvergencia szintre. Ha nem értük el a konvergencia szintet, akkor az előző két mátrixot felcserélve iterálunk tovább.

IV-A4. Adatok mentése: Tesztelhetőségi megfontolások végett a nem csak a türe ható erőt (villamos térerősséget) exportáljuk, hanem a konvergencia szintjének változását és az interpolált felületet is. Az exportálandó menysícek könnyen kézben tartható mérete miatt egyszerű CSV fájlként kerülnek mentésre. Ezen fájlok további poszt-processzálása MATLAB avagy munkalap kezelő szoftverrel is elvégezhető.

IV-B. OpenCL architektúrája

Az Open Computing Language (OpenCL) keretrendszer [6] közös nyelvet, magas szintű programozási interfészt és hardware absztrakciót nyújt a fejlesztőknek adat- vagy feladat párhuzamos számítások gyorsítására különböző számítógépsíven (CPU, GPU, FPGA, DSP, ...). Az OpenCL modellje a különböző „device”-okat, amik több „compute unit”-ot (processzor-magot) tartalmaznak heterogén módon kezel.



5. ábra. OpenCL „device” architektúra [6]

A „compute unit”-ok kiéheztetésének elkerülése végett (több ezer) „work-item” virtuális osztozik rajta. Továbbá ezen „work-item”-ek „work-group”-okba vannak rendezve, később részletezett megfontolások végett. A „compute unit” kiéheztetését a „device”-on található memória chipek lassúsága okozza. Ennek hárdiveres megoldása a több szintű prediktív cache memória beiktatása a „compute unit” és a külső

I. táblázat. OpenCL memória szintek

	Global	Constant	Local	Private
Host	Dinamikus R/W	Din. R/W	Din. R/W	
Kernel	R/W	Statikusan R	Satik. R/W	Statik. R/W
Sebesség	Lassú	Gyors	Gyors	Regiszter
Méret	1 Gbyte <	~ 64 Kbyte	~ 16 Kbyte	< 1 Kbyte

II. táblázat. nVidia GeForce 330M OpenCL tulajdonságai

MAX_COMPUTE_UNITS	6
MAX_WORK_GROUP_SIZES	512 512 64
GLOBAL_MEM_SIZE	1073020928
MAX_CONSTANT_BUFFER_SIZE	65536
LOCAL_MEM_SIZE	16384

memória közé. Mivel a bank szervezett külső memóriák hozzáférési ideje relatíve nagy így a memória szervezésére nagy hangsúlyt kell fektetni.

Az OpenCL négy memória szintet különböztet meg, ami az I táblázatban és az 5. ábrán látható. Ahhoz, hogy a rendszerben rejlő teljesítményt kihozzuk három fontos kérdést kell a szimulátor magjának implementálásakor megválaszolnunk:

- **Mennyit?:** Tisztában kell lennünk az aktuális memória fogyasztással és a szükséges memóriamérettel.
- **Honnan-hova?:** Fontos, hogy a lehető legközelebb legyen az adat a „work-item”-hez.
- **Mikor?:** Mivel a memória művelet alatt a „work-item” nem dolgozik, így átadja a helyét egy másiknak. Ennek a megfelelő szinkronizációjával nagyobb kihasználtság érhető el (load balance).

OpenCL keretrendszerben történő programozás során két programot kell írunk. Az egyik a „host”-on fut, ami elvégzi a probléma összeállítását, memória allokálását, argumentumok beállítását és a másik program a kernel meghívását a „device”-on. A kernel futása végeztével a „host” program kiolvassa a „device”-ből a kívánt eredményt.

IV-C. Implementációhoz szükséges megfontolások

A következőkben egy kisebb teljesítményű notebook videokártyát veszek alapul a megfontolások demonstrálására. Ez az nVidia GeForce 330M, 575 MHz-en futó 48 CUDA core-al, 1024GB memóriával és OpenCL 1.0 kompatibilitással. A videokártya továbbiakban fontos paraméterei a II. táblázatban látható.

Ha a tér ahol a laplace egyenletet meg kell oldanunk nagyon nagy, akkor érdemes szétbontani kisebb alterekre és azokhoz rendelni egy-egy „work-item”-eket. Mivel a diszkrét laplace egyenlet egy pontja a szomszédos pontokkal szoros kapcsolatban van, így az összefüggő „work-item”-eket egy „work-group”-ba érdemes szervezni, mivel így az átlapolódó pontok értékét a szomszédos „work-item” is tudja írni és olvasni. Az ilyen típusú problémának méretét a MAX_WORK_GROUP_SIZES tulajdonság korlátozza.

Jelen esetben a mérési eredmény egy pontjához tartozó tér átlagosan $11 \times 11 \times 30$ pontból áll. Tehát a korábbi nem áll fenn és egyszerű megfeleltetéssel szétoszthatjuk a feladatot. A teljes tér $512 \times 512 \times 11 \times 11 \times 30$ méretű, ami 951k pont. A tárolásához single-precision mellett ennek a számnak a 4-szerese szükséges byte-okban mérve. Mivel ez a videokártyán nem áll rendelkezésre, így szétbontjuk kisebb feladatrészekre.

Ezen feladatrészek méretét egy paraméter állításával lehet változtatni és az implementált algoritmus ettől generikusan függ. Emellett

az interpoláció mértéke is generikusan paraméterrel állítható. Az algoritmus generikusságát csupán a futási időben történő dinamikus memória allokációval lehetséges megvalósítani. A korábban említettek végett (I táblázat) az allokáció csak a „host” programban történhet.

IV-D. Memória szervezés

IV-D1. Csak globális memória használata: Az algoritmus pszeudó kódjának direkt leképezése esetén a „host”-on allokálunk memóriát a „device” globális memóriájában. Majd a megfelelő adatokat ide másoljuk és a kernel is itt ír és olvas. A problémát a globális memória nagy hozzáférési ideje jelenti, ami miatt sok „work-item” tétlenül a memóriára fog várakozni. Ilyenkor az egy mérési pontra vonatkoztatott szimulációs idő a referenciánál is lassabb.

IV-D2. Globális memória és adott esetben lokális memória használata: Kis erőfeszítéssel nagy javulást lehet elérni, ha a mérési ponthoz tartozó szimulációs tér éppen belefér a lokális memóriába. Tehát, mielőtt az ?? szerinti iteratív megoldót futtatnánk először a globális memóriából a lokális memóriába töltjük át a kérdéses pontokat, majd számolunk rajt és a végén visszatöltjük a globális memóriába. E javítással a referenciával azonos sebességet tudunk elérni.

IV-D3. Globális memória és minden adódó alkalomkor a lokális memória használata: Nagyobb erőfeszítést igényel, hogy a globális memóriával való kommunikációt a lokális memória közbeékelésével tegyünk minden alkalomkor. Ezt úgy lehet felfogni, mintha a globális memóriát lokális memória méretű kvantumokban tudnánk csak elérni. Ekkor nagy odafigyelést kíván a memóriacímzés megfelelő prgramozása, de eredményképp gyorsulás elérhető.

Összegezve elmondható, hogy az aktuálisan használt adat tárolását a lehető legközelebb kell tartani a „compute-unit”-hoz.

V. EREDMÉNYEK

V-A. MATLAB implementációk

A referenciaként szolgáló MATLAB algoritmus lineáris program-szervezést alkalmazva az elérhető fajlagos futási idő $\sim 100ms$.

A kód minimális változtatásával elérhető a párhuzamos végrehajtás. Ezt a for ciklusok Parallel Toolbox belüli parfor utasítására cserélve érhetjük el. 4 processzormaggal rendelkező PC esetén ilyenkor közel a gyeredére csökken a futási idő.

V-B. OpenCL implementációk

OpenCL keretrendszer segítségével írt programot a GPU-n futtatva a III táblázatban látható eredményeket kapjuk. Csupán a globális memóriát használva a referenciához képest romlik a teljesítmény. Ezt a videokártya prediktív cache nélküli kialakításának és a globális memóriája okozta kiéheztetésnek tudhatjuk be. A lokális memória használata a futási időt drasztikusan le tudja csökkenteni, ami a korábban ismertetett memória szervezési gondolatok helyességét igazolja.

VI. ÖSSZEGZÉS

A cikkünkben ismertettük az AFM mérések validálásának problémáját. A szimulálandó problémát ismertette adódott a párhuzamosítás lehetősége. A hordozhatóság és a gyorsabb végrehajtás végett a szimulátort OpenCL környezetben implementáltuk. Ismertettük az implementáció során észben tartandó gondolatokat, azok általánosságának elvesztése nélkül.

Az eredményeket ismertette látványos gyorsulást tapasztaltunk, ami nehezebben kezelhető problémáknál azonos futási idő mellett nagyobb pontosságot jelenthet.

HIVATKOZÁSOK

- [1] G. Binnig, C. F. Quate, and C. Gerber, „Atomic force microscope,” *Phys. Rev. Lett.*, vol. 56, pp. 930–933, Mar 1986. [Online]. Available: <http://link.aps.org/doi/10.1103/PhysRevLett.56.930>
- [2] B. Vasić, M. Kratzer, A. Matković, A. Nevesad, U. Ralević, D. Jovanović, C. Ganser, C. Teichert, and R. Gajić, „Atomic force microscopy based manipulation of graphene using dynamic plowing lithography,” *Nanotechnology*, vol. 24, no. 1, p. 015303, 2013. [Online]. Available: <http://stacks.iop.org/0957-4484/24/i=1/a=015303>
- [3] H.-J. Butt, B. Cappella, and M. Kappl, „Force measurements with the atomic force microscope: Technique, interpretation and applications,” *Surface Science Reports*, vol. 59, no. 1–6, pp. 1 – 152, 2005. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0167572905000488>
- [4] Y. Martin, C. C. Williams, and H. K. Wickramasinghe, „Atomic force microscope-force mapping and profiling on a sub 100 a scale,” *Journal of Applied Physics*, vol. 61, no. 10, pp. 4723–4729, 1987. [Online]. Available: <http://scitation.aip.org/content/aip/journal/jap/61/10/10.1063/1.338807>
- [5] S. Hudlet, M. Saint Jean, C. Guthmann, and J. Berger, „Evaluation of the capacitive force between an atomic force microscopy tip and a metallic surface,” *The European Physical Journal B - Condensed Matter and Complex Systems*, vol. 2, no. 1, pp. 5–10, 1998. [Online]. Available: <http://dx.doi.org/10.1007/s100510050219>
- [6] Khronos OpenCL Working Group, *The OpenCL Specification, version 1.0*, 6 August 2010. [Online]. Available: <http://www.khronos.org/registry/cl/specs/opencl-1.0.pdf>

III. táblázat. OpenCL futási idő eredmények 12×12 mérési pontra

	Globális memória	Lokális memória, ha befér	Lokális memória bufferelés
Globális tranzakciók száma átlagosan	$12 \times 12 \times 32.3$	$12 \times 12 \times 32.3$	$12 \times 12 \times 32.3$
Lokális tranzakciók száma átlagosan	0	$0.48 \times 12 \times 12 \times 30$	$2.08 \times 12 \times 12 \times 32.3$
Futási idő	5990 ms	2530 ms	510 ms
Fajlagos futási idő	410 ms	170 ms	3.5 ms



(a) Mérési eredmény



(b) Szimulációs eredmény

6. ábra. Méréssel kapott felület (balra) és szimulációval kapott töltéstérkép