
**Budapesti Mûszaki és Gazdaságtudományi Egyetem**
Villamosmérnöki és Informatikai Kar
Szélessávú Hírközlés és Villamosságtan Tanszék

# Poros plazma kísérletek támogatása multiprocesszoros környezetben

## Diplomatervezés 1. - beszámoló

*Készítette*
Bakró Nagy István

*Konzulens*
Hartmann Péter & Reichardt András

2014. április 26.

# Tartalomjegyzék

# FELADATKIÍRÁS

A modern alacsony hőmérsékletű plazmafizikai kísérletek egy új, érdekes és izgalmas területe a poros plazmák kutatása. Egy elektromos gázkisülésbe helyezett apró (mikrométer méretű) szilárd szemcse a kisülési plazma atomi részecskéivel kölcsönhatva elektromosan feltöltődik. A sok töltött szemcséből kialakuló elrendezésben a szilárdtestfizikai jelenségek széles spektruma figyelhető meg, pl. kristályrács kialakulása, fázisátalakulás, diszlokációk dinamikája, transzport folyamatok, stb. Poros plazmákat jelenleg leginkább alapkutatásokban alkalmaznak, de jelentőségük az elektronikai gyártásban, fúziós reaktorok üzemeltelésében, Teraherz technológiában egyre inkább előtérbe kerül.

A kísérleti adadtgyűjtés és feldolgozás nagyrésze részecske-követő velocimetrián (particle tracking velocimetry) alapul, vagyis első lépésben egy nagysebességű kamera segítségével nagyfelbontású képek készülnek, amely képek segítségével a porszemcsék pontos (a kamera felbontásánál pontosabb) koordinátáit kell meghatározni. A képek elemzése ezidáig csak a mérést követően, hosszú idő alatt volt megvalósítható a vizsgálandó nagy adatmennyiség miatt. A multiprocesszoros környezetek segítségével a feldolgozás gyorsítása lehetséges akár több nagyságrenddel is.

A jelölt feladata, hogy a meglévő kísérleti elrendezés, amely az MTA Wigner Fizikai Kutatóközpont Szilárdtestfizikai és Optikai Intézezetben található, kiegészítésével a mérés közbeni feldolgozással a mérést segítő analízist hajtson végre. Ennek eredményével a mérés előkészítése és elvégzése lényegesen gyorsulhatnak.

**A jelölt feladata**

- Mutassa be a mérési elrendezést és elemezze a kapott adatokat! (Mutassa be a mérést!)
- Elemezze a lehetséges multiprocesszoros környezeteket, a feladat szempontjából lényeges paraméterek és feladatvégrehajtási elvárások szempontjából!
- Készítsen programot, amely az azonnali (valós idejű) analízisben résztvevő paramétereket számítja ki, a multiprocesszoros környezet kihasználása nélkül!
- Készítsen programot, amely a mérési környezetbe illeszkedve a mérésnél valós időben képes a vizsgált paraméterek megjelenítésére! Mutassa be és elemezze az elkészített programot!
- Hasonlítsa össze a multiprocesszoros és a nem-multiprocesszoros környezetre elkészített programokat erőforrás igény illetve egyéb paraméterek szempontjából!

**Irodalom**:


- [1] Hartmann P, et. al. ; "Crystallization Dynamics of a Single Layer Complex Plasma"; Phys. Rev. Lett., 105 (2010) 115004
- [2] Hartmann P, et. al. ; "Magnetoplasmons in Rotating Dusty Plasmas", Phys. Rev. Lett. 111, 155002 (2013)
- [3] Hartmann P, Donkó I, Donkó Z; "Single exposure three-dimensional imaging of dusty plasma clusters"; Rev. Sci. Instrum., 84 (2013) 023501/1-5;


**Tanszéki konzulens**: Reichardt András, egy. tanársegéd
**Külső konzulens**: Hartmann Péter, PhD., tud. főmunkatárs (MTA Wigner FK, SZFI)

Budapest, 2014.03.10.

# Kivonat

hablaty

# 1. fejezet

# A mérés

## 1.1. A mérési elrendezés

$$W(s) = \frac{A}{1 + 2T\xi s + s^2T^2}. \tag{1.1}$$

## 1.2. A mérendő mennyiségek és a származtatott értékek

# 2. fejezet

# A multiprocesszoros környezet

## 2.1. Architektúra bemutatása

## 2.2. Megfontolandó gondolatok

## 2.3. Futási környezet bemutatása

**2.1. táblázat.** *Futási környezetek*

| | nVidia GT330M | Xeon PHI |
|---|---|---|
| COMPUTING_UNITS | 6 | 256 |
| GLOBAL_MEMORY_SIZE | 1 Gbyte | 4 Gbyte |

# 3. fejezet

# A referenciaként szolgáló program

## 3.1. A program bemutatása

### 3.1.1. Detektálási algoritmus

### 3.1.2. Mennyiségek származtatása

## 3.2. Implementáció során hozott döntése

# 4. fejezet

# Multiprocesszoros program

## 4.1. Fejlesztőkörnyezet összeállítása

OpenCL kód fejlesztése történhet Windows alatt NVIDIA Nsight Visual Studio Edition [4] és Linux alatt GCC-vel [5]. Az Open Source fejlesztőrendszer ingyenessége és az általa generált program hordozhatósága végett a Linux alatti fejlesztés mellett döntöttem. Az OpenCL-t támogató hardverek legtöbbször CPU-k, GPU-k és az Intel MIC [6] kártyái. Ezekre való OpenCL kód fejlesztéséhez a gyártók biztosítanak Software Developement Kit-et (SDK). Ezek telepítése szinte bármelyik Linux disztribúción sikerülhet a megfelelő követelmények előzetes telepítése után. A Linux diszrók közül a CentOS-re [7] esett a választás, ami csupán a fejlesztőkörnyezet egyszerűbb telepítése végett történt így.

### 4.1.1. Software Developement Kit-ek (SDK) telepítése

#### nVidia támogatás telepítése

A legtöbb mai Linux disztrók tartalmaznak drivert az nVidia videó kártyákhoz. Ez az open source Nouveau, ami még nem támogatja az OpenCL-t. Így a hivatalos nVidia drivert fel kell telepítenünk. Ehhez először le kell tiltanunk a Nouveau betöltését. Ezt két helyen is meg kell tennünk: egyrészt a `/etc/modprobe.d/blacklist.conf` fájlhoz hozzá kell adnunk a következő sort:

```
blacklist nouveau
```

majd újragenerálni az INITial RAM File System-et (initramfs), ami a rendszer inicializásáért felelős:

```
$ mv /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r).img.bak
$ dracut -v /boot/initramfs-$(uname -r).img $(uname -r)
```

másrészt a rendszer indító GRand Unified Bootloader-ben (GRUB) is le kell tiltani a betöltését a kernel opció alábbi paranccsal való kiegészítésével:

```
nouveau.modeset=0
```

Továbbá a telepítéshez szükséges követelményeket a következő parancsokkal telepíthetjük:

```
$ yum groupinstall "Development Tools"
$ yum install kernel-devel kernel-headers dkms
```

Ekkor a rendszer újraindítása után készen állunk a hivatalos nVidia driver telepítésére. A drivert a következő linken lehet letölteni [8]. A grafikus felületet a telepítés idejére le kell állítani az X grafikus kiszolgálót

```
$ init 3
```

paranccsal, majd a konzolban telepíthető a driver, ami a legtöbb munkát elvégzi helyettünk. Ezután az

```
$ init 5
```

paranccsal áttérhetünk a grafikus felületre, ahol a megfelelő környezeti változókat kiegészíthetjük. Legcélratörőbb, ha a `~/.bashrc` fájlt módosítjuk és hozzáadjuk a következő sorokat:

```
PATH=$PATH:$HOME/bin:/usr/local/cuda/bin
export PATH

CUDA_INSTALL_PATH=/usr/local/cuda
export CUDA_INSTALL_PATH

LD_LIBRARY_PATH=/usr/local/cuda/lib64:/opt/intel/opencl/bin
export LD_LIBRARY_PATH

NVSDKCOMPUTE_ROOT=/usr/local/cuda/lib64
export NVSDKCOMPUTE_ROOT

INTELOCLSDKROOT=/opt/intel/opencl
export INTELOCLSDKROOT
```

Mivel az nVidia limitálja a kernel futási időt 5 másodpercben limitálja, hosszabb kernel futási idő esetén a rendszer lefagy. Ezt a korlátozást a `/etc/X11/xorg.conf` fájl `Device` részének a következővel való kiegészítésével érhetjük el:

```
Option "Interactive" "boolean"
```

Érvényre juttatásához az X újraindítása szükséges (CTRL+ALT+Backspace). Ezután nagyobb problémák esetén már nem fogja lefagyasztani a rendszert a watchdog.

### Intel támogatás telepítése

A következő oldalról letölthetjük az SDK-t [9]. A kicsomagolás után az `./install-cpu.sh` program futtatásával telepíthető. Ezután még szükséges a `LD_LIBRARY_PATH` beállítása.

### 4.1.2. Eclipse – Integrated Developement Environment

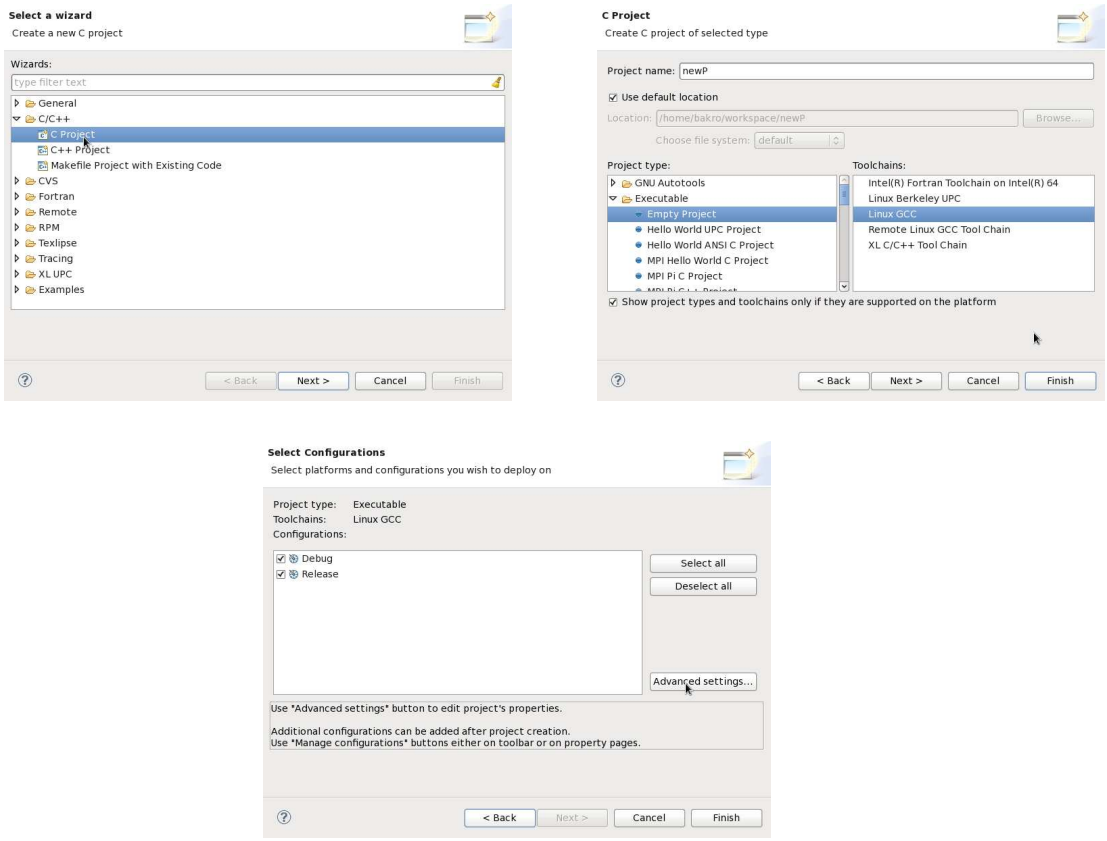
A fejlesztés és hibakeresés egy Integrated Developement Enviroment (IDE) segítségével könnyebb. Az open source Eclipse [10] fejlesztőkörnyezet a különböző pluginjaival épp megfelelő erre a célra. Például a C-nyelv fejlesztését segítő C/C++ Developement Tooling (CDT), a verziókövetést menedzselő EGit és a hibakeresést támogató GDT. A sok Eclipse változat közül az OpenCL fejlesztéshez legjobban az Eclipse for Parallel Application Developers verzió illik, mivel a korábban említett pluginokat már eleve tartalmazza.

## 4.2. Új (Hello World) projekt létrehozása

Az OpenCL fejlesztését konyhanyelven bemutató OpenCL Programming Guide [11] könyvben szereplő Hello World programot a következő linken lehet letölteni [12]. A kód fordítása elött egy Eclipse projektet létrehozunk és a fordításhoz szükséges beállításokat elvégezzük.

### 4.2.1. Empty C project létrehozása

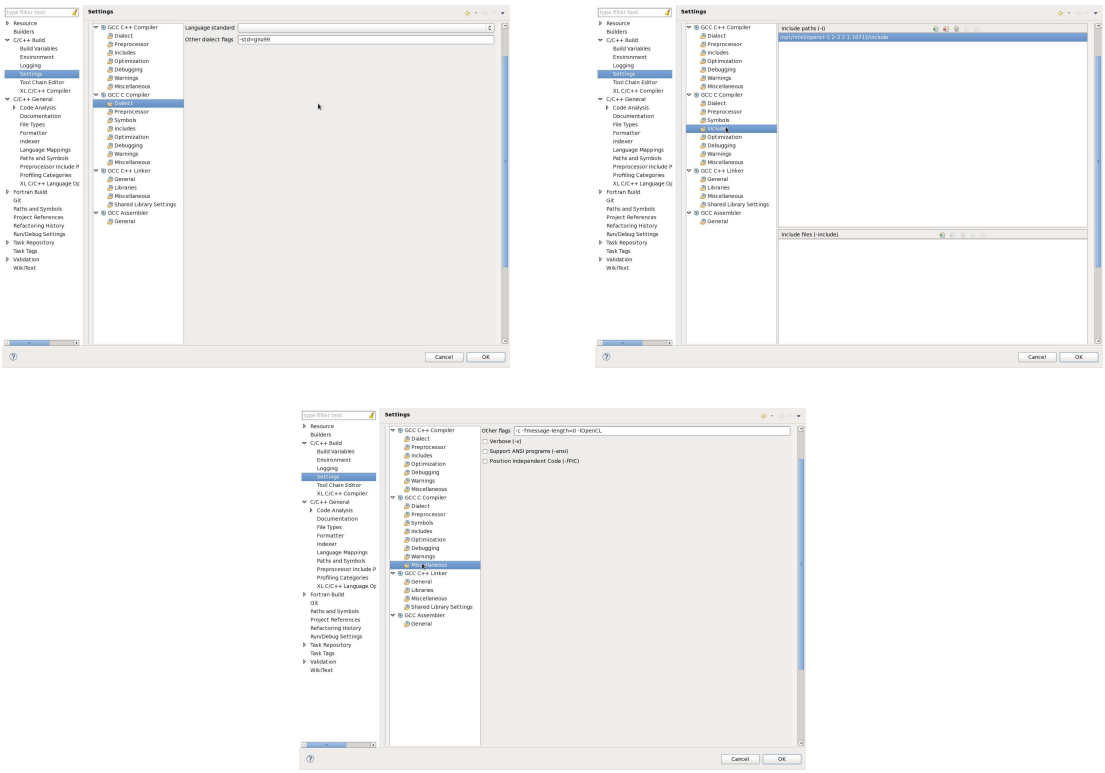
Először egy üres C projektet hozunk létre, ami folyamatát a 4.1 ábrán látjuk. A korábban említettek szerint fordítónak a Linux GCC-t állítjuk be.




**4.1. ábra.** *Új Eclipse projekt létrehozása*

### 4.2.2. Compiler beállítása

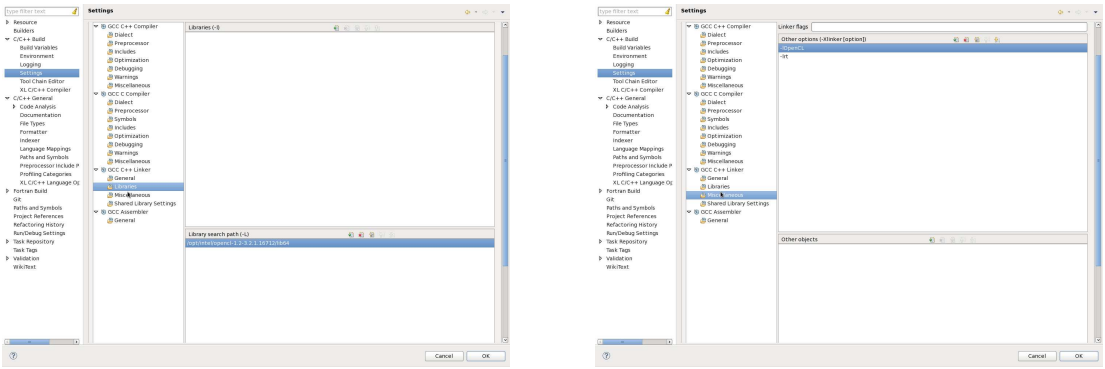
A létrehozott projektre jobb gombbal kattíntva a tulajdonságára kattintva állíthatjuk be a fordítót a képnek 4.2 megfelelően. A beállítások kiterjednek a GNU-C99 nyelv szerinti fordításra és a korábbi 4.1.1 részben telepített SDK-ban található include mappa beállítására.



**4.2. ábra.** *Compiler beállításai*

### 4.2.3. Linker beállítása

A linkert a 4.3 ábra szerint állítjuk be.



**4.3. ábra.** *Linker beállításai*

## 4.3. Kód

**4.1. lista.** *A fenti számozott felsorolás LaTeX- forráskódja*

```
\begin{enumerate}
        \item \emph{e l s   bajusz:} ide lehetne  rni  az e l s  elem k i f e j  s  t ,
         s  ez a k i f e j t  s  gy  n  z  ki, ha  t  bb sorosra sikeredik,
        \item \emph{m sodik bajusz:} ide lehetne  rni  a m sodik elem k i f e j  s  t ,
        \item \emph{ez meg egy szak l l :} ide lehetne  rni  a harmadik elem k i f e j  s  t .
\end{enumerate}
```

# 5. fejezet

# Összehasonlítás

**5.1. táblázat.** *Az órajel-generátor chip órajel-kimenetei.*

| Órajel | Frekvencia | Cél pin |
|---|---|---|
| CLKA | 100 MHz | FPGA CLK0 |
| CLKB | 48 MHz | FPGA CLK1 |
| CLKC | 20 MHz | Processzor |
| CLKD | 25 MHz | Ethernet chip |
| CLKE | 72 MHz | FPGA CLK2 |
| XBUF | 20 MHz | FPGA CLK3 |

# 6. fejezet

# Összegzés

Dolgozatomban ...

# Ábrák jegyzéke

# Táblázatok jegyzéke

# Irodalomjegyzék


[1] P. Hartmann, A. Douglass, J. C. Reyes, L. S. Matthews, T. W. Hyde, A. Kovács, and Z. Donkó, „Crystallization dynamics of a single layer complex plasma," *Phys. Rev. Lett.*, vol. 105, p. 115004, Sep 2010.

[2] P. Hartmann, Z. Donkó, T. Ott, H. Kählert, and M. Bonitz, „Magnetoplasmons in rotating dusty plasmas," *Phys. Rev. Lett.*, vol. 111, p. 155002, Oct 2013.

[3] D. Z. Hartmann P, Donkó I, „Single exposure three-dimensional imaging of dusty plasma clusters," *Rev. Sci. Instrum.*, vol. 84, p. 023501, 2013.

[4] „Nvidia nsight visual studio edition." `https://developer.nvidia.com/nvidia-nsight-visu`

[5] „Gcc, the gnu compiler collection - gnu project - free software foundation (fsf)." `http://gcc.gnu.org/`.

[6] „Intel® many integrated core architecture (intel® mic architecture)." `http://www.intel.com/content/www/us/en/architecture-and-technology/many-integ`

[7] „Centos project." http://www.centos.org/.

[8] „Nvidia drivers." `http://www.nvidia.com/Download/index.aspx`.

[9] „Intel opencl sdk." `https://software.intel.com/en-us/vcsource/tools/opencl-sdk-xe`

[10] „Eclipse - the eclipse foundation open source community website.." `https://www.eclipse.org`.

[11] A. Munshi, *OpenCL Programming Guide*. Addison-Wesley, 2011.

[12] „Opencl programming guide - examples." `https://code.google.com/p/opencl-book-sampl`