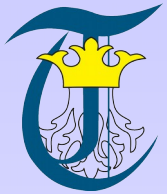


Universitatea *Transilvania* Brașov
Facultatea de Inginerie Electrică și Știința Calculatoarelor
Departamentul de Electronică și Calculatoare

Sisteme de operare distribuite și în timp real

– curs –
master SECI anul I

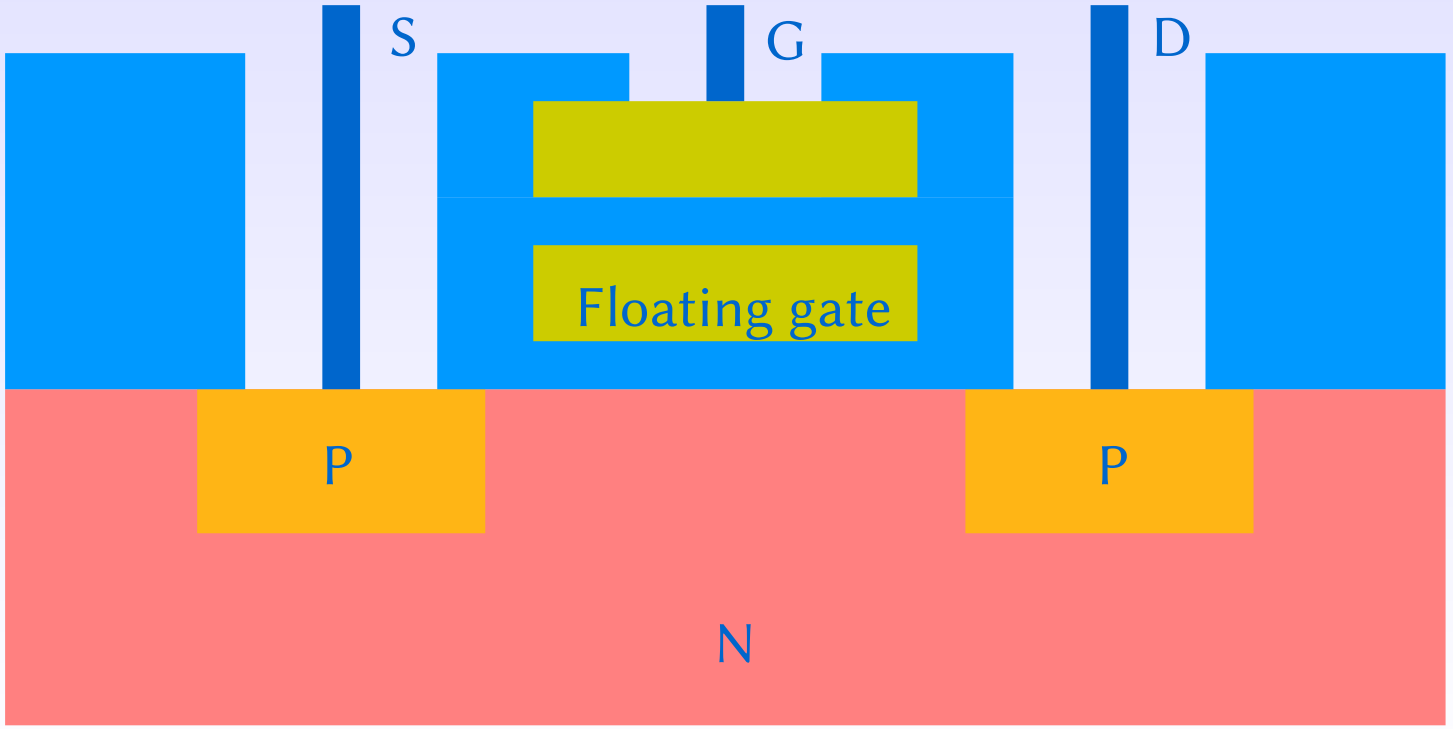
ș.l. dr. ing. Kertész Csaba-Zoltán

Stocarea datelor în sisteme încorporate

- stocarea programelor și datelor trebuie realizată în memorii nevolatile:
  - ROM
    - foarte ieftin în serie mare
    - nu mai poate fi reprogramat
  - NVRAM
    - viteză mare de lucru: ideal pentru stocarea datelor
    - foarte scump
  - Flash
    - densitate mare, relativ ieftin
    - reprogramabil

Memorii Flash

- dezvoltat în anii '80 de Toshiba și Intel
- folosește tranzistori MOS cu o poartă de control și o poartă flotantă



Operaţii asupra celulelor Flash

- poarta flotantă poate fi încărcată sau nu
  - încărcarea normală nu este suficient pentru deschiderea canalului DS
  - fiind complet izolată sarcinile acumulate pe poarta flotantă se păstrează pentru timp îndelungat
- citire:
  - se aplică o tensiune (jumătate din cea necesară deschiderii canalului) pe poarta de control
  - tensiunea aplicată împreună cu sarcinile stocate pe poarta flotantă este suficientă pentru a deschide canalul DS

Operații asupra celulelor Flash

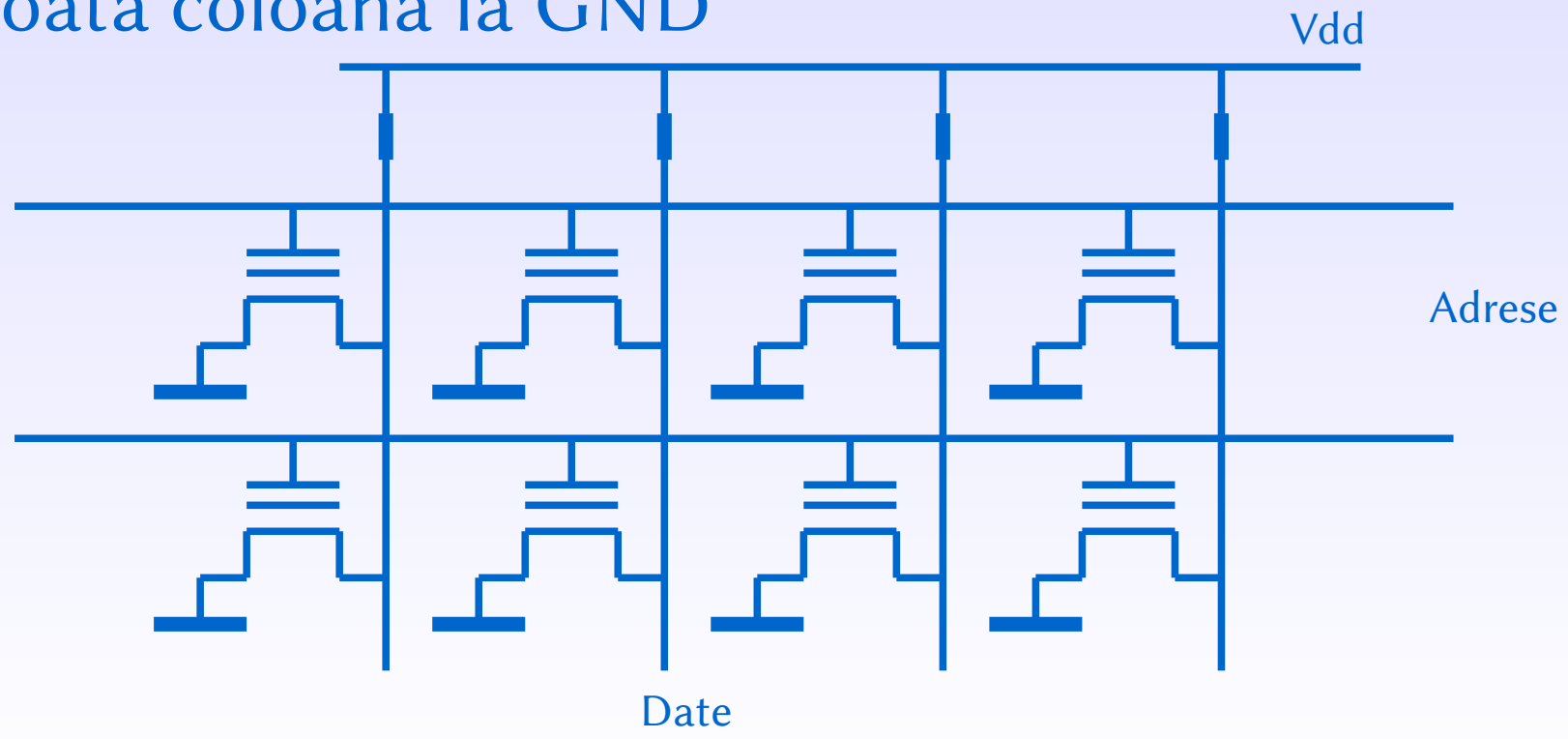
- scrierea (injecție de electroni fierbinți)
  - se aplică o tensiune mare pe poarta de control (mult mai mare decât tensiunea de deschidere)
  - datorită curentului mare pe canalul DS deschis, electronii „fierbinți” sar pe poarta flotantă încărcând aceasta
- ștergerea (efectul tunel)
  - se aplică o tensiune negativă mare pe poarta de control
  - sarcinile stocate pe poarta flotantă se vor extrage prin efectul tunel

Organizarea celulelor Flash

- există două tehnologii (denumite după modul de legare a tranzistorilor):
  - NOR (sau negat cablat)
  - NAND (și negat cablat)
- organizarea în amândouă cazuri este una matricială cu linii de adresă care vor selecta celulele paralele

Flash NOR

- tranzistorii sunt legați pe o parte la masă pe cealaltă parte pe coloana corespunzătoare
- dacă unul din tranzistori este deschis se va trage toată coloana la GND

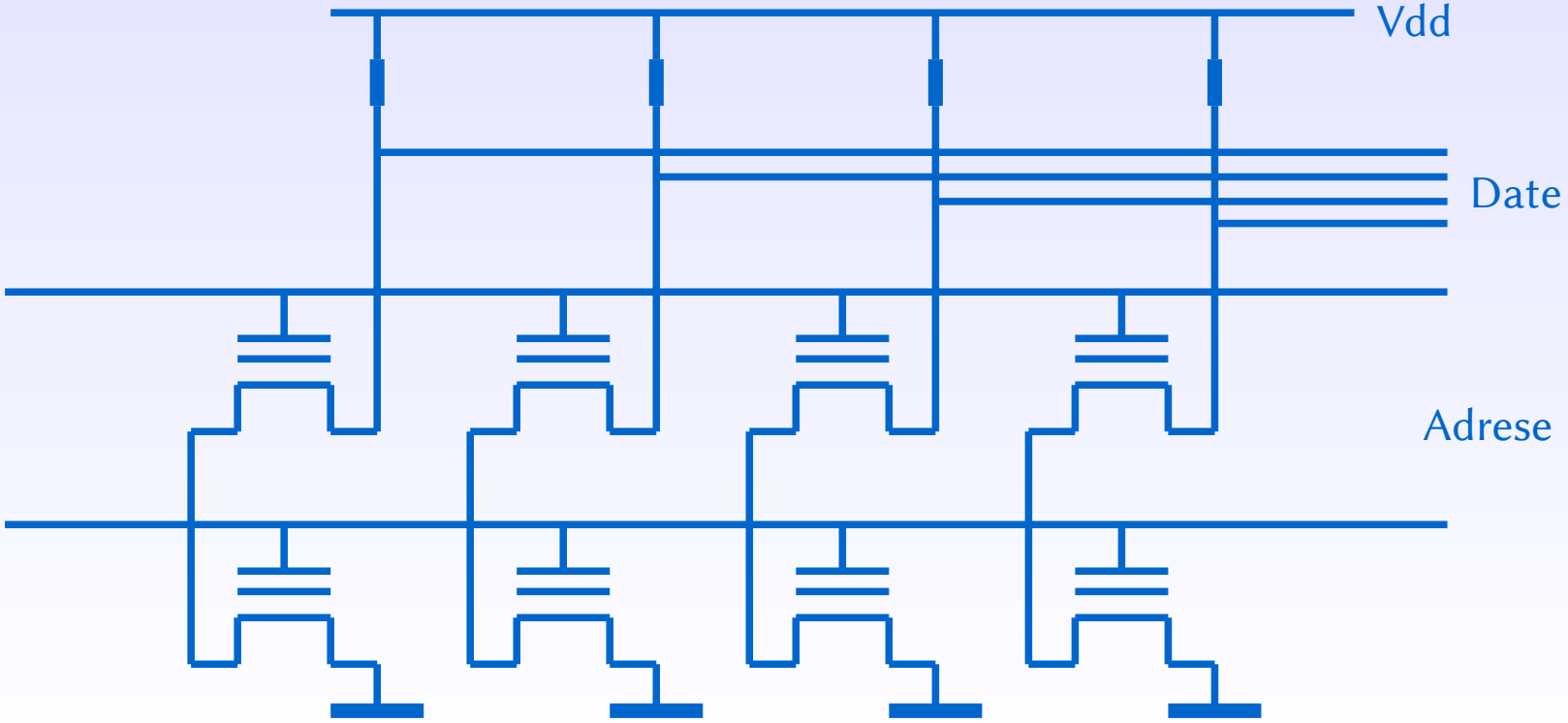


Flash NOR

- celule sunt organizate în blocuri
  - dimensiunea blocului este de ordinul 64k
- citirea și scrierea se poate face prin adresare directă
  - liniile de adrese și date sunt organizate similar cu memoriile statice
- ștergerea trebuie făcută pe bloc
- tensiunile de scriere și ștergere sunt realizate intern prin convertoare boost integrate

Flash NAND

- tranzistorii de pe o coloană sunt legați în serie
- se activează toate liniile de adresă cu tensiunea normală, mai puțin adresa selectată (pe care se aplică jumătatea de tensiune)



Flash NAND

- celule sunt organizate în pagini, iar paginile în blocuri
- paginile trebuie citite și scrise deodată, ștergerea se face pe blocuri
- conține buffer SRAM integrat pentru pagina citită care poate fi citit ca un flux de date pe interfața externă
- comunicarea externă se bazează pe comenzi către registrele interne

NOR vs. NAND
Acces la date

- citire/scriere de la orice adresă
- ștergere pe blocuri
- interfață cu CPU similar cu SRAM (magistrală de control, adrese și date)
- citire foarte rapidă, scriere, ștergere lentă

- citire/scriere câte o pagină deodată
- ștergere pe blocuri
- protocol de comunicație special, comenzi către registre de adrese și date interne
- citire, scriere, ștergere rapidă

NOR vs. NAND
Aplicații

- fiind accesibil aleator la nivel de octet / cuvânt, poate fi folosit pentru rularea codului direct din flash
- este folosit pe post de memorie nevolatilă integrată în microcontrolere și pe post BIOS/bootloader extern
- lucrează în mod similar ca și harddiskul (accesare pe blocuri, cu controler integrat
- este folosit pe post de dispozitiv de stocare de înaltă capacitate (SSD, card, stick)

NOR vs. NAND
Fiabilitate și costuri

- foarte scump datorită suprafeței mari ocupate și standardelor ridicate de fabricare
- sectoare bad practic inexistente

- mai ieftin datorită posibilității de integrare mai mare și proceselor tehnologice mai permisive
- pot să existe sectoare bad (fiind proiectate în scopul înlocuirii hard-diskurilor)

Caracteristici specifice flashurilor

- durata de viață (calculată în numărul de ștergeri) este limitată
- pentru a crește durata de viață ștergerile de blocuri ar trebuie echilibrate în tot spațiul de flash (wear leveling)
- operație de rescriere este complicată
  - modificarea unui bit de 1 în 0 se poate face la nivel de cuvânt (sau pagină)
  - modificarea unui bit de 0 în 1 necesită ștergerea completă a unui bloc întreg după care vor trebui rescrise și toate datele nemodificate

Folosire flashului în Linux

- modul tradițional de realizare a driverelor și a sistemelor de fișiere în Linux nu poate fi adaptat ușor la memoriile flash
  - acestea pot fi citite pe caractere, dar trebuie scrise pe blocuri
  - sistemele de fișiere din Linux nu oferă suport specific flashurilor
    - nu există wear leveling
    - blocurile din flash sunt de obicei mult mai mari decât sectoarele folosite pe hard-diskuri
    - folosirea cacheului pentru file system trebuie evitat pentru a nu avea probleme la oprirea bruscă a sistemului

MTD

- inițial flashurile se integrau în Linux printr-un FTL (Flash Translation Layer) care emula un hard-disk
  - nu soluționa toate problemele legate de caracteristicile flashurilor
  - nu ofera suport pentru alte moduri de operare
  - devenea din ce în ce mai complex cu apariția noilor tipuri de flashuri și noilor interfețe
- soluția era introducerea subsystemului MTD (Memory Technology Devices)

MTD

- MTD a fost conceput special pentru modul de operație cu flashuri
- se integrează în Linux atât pe nivelul de drivere cât și pe nivelul de aplicații
- la nivelul de drivere MTD mapează pentru fiecare dispozitiv câte un driver de character și de block device
  - character device poate fi folosit în mod obișnuit cu funcțiile `open`/`read`/`write`/`ioctl`
  - block device poate fi folosit pentru montarea unui sistem de fișiere clasic

Arhitectura MTD

- MTD core
  - implementează driverele character și block
- drivere flash low-level
  - implementează interfețele pentru flashuri NOR și NAND
- BSP pentru flash
  - oferă suport pentru modurile de conectare a flashului pe placă
- aplicații MTD
  - aplicații pe nivel kernel care folosesc flash (JFFS2) și aplicatii pe nivel de utilizator (upgrade)

Suport MTD în embedded Linux

- MTD pentru flashuri NOR a fost introdus în kernelul 2.4,
- extensia pentru flashuri NAND (numai pentru NFTL – NAND Flash Translation Layer) a fost introdus în versiuni ulterioare a kernelului 2.4, și suport complet în kernelul 2.6
- există implementare completă (drivere low level) pentru flashuri NOR cu interfața CFI și pentru dispozitive flash NAND uzuale (SSD, SD/MMC/MS/CF/xD)

Maparea memoriilor Flash

- sistemele embedded de obicei pornesc execuția dintr-o memorie flash (de tip NOR)
- poate exista și memorie flash NAND în sistem, care trebuie montat ca un hard-disk, și transferat conținutul în memorie pentru a putea fi executat
- Linux are nevoie de un sistem de fișiere montat pentru /, care poate fi un flash NAND sau o partiție în memorie
  - aceasta înseamnă că în sistemul încorporat trebuie să existe cel puțin două partiții (boot și /)

Sisteme de fișiere embedded

- fiindcă Linux presupune existența unui sistem de fișiere chiar și în lipsa hard-diskurilor, este nevoie de folosirea unor sisteme de fișiere dedicate (care folosesc fie memoria RAM fie o memorie flash):
  - Ramdisk
  - RAMFS
  - CRAMFS
  - JFFS2
  - NFS

Ramdisk

- este o metodă de emulare a unui hard-disk în memorie
- nu este un sistem de fișiere, ci mai degrabă o tehnologie ce permite crearea sistemului / în memorie, după care în acest sistem se pot monta alte sisteme de fișiere
- de obicei initrd se crează într-o formă de imagine ramdisk, ce poate fi încărcată în memorie de către boot-loadeer

RAMFS

- este un sistem de fișiere localizată în întregime în memorie RAM
- de multe ori este nevoie de folosirea unor informații (interne ale kernelului sau a utilizatorilor) organizate în formă de fișiere pentru un acces comod și care nu trebuie păstrate după repornirea sistemului, de aceea nu merită a fi încărcate în flash
- RAMFS oferă suport pentru stocare a astfel de informații într-o zonă de memorie care își poate modifica dimensiunea în funcție de necesități

CRAMFS

- este un sistem de fișiere read-only cu compresia datelor
- este foarte util pentru stocarea programelor în flash comprimate pentru a minimiza spațiul folosit
- CRAMFS este un sistem de fișiere normal, care comunică prin drivere block folosind un cache în memorie și algoritmi de compresie din zlib pentru pagini de 4kB

JFFS2

- sistem de fișiere cu jurnal special destinat pentru flashuri
- realizează wear leveling
- nu folosește cache de scriere pentru a evita coruperile de date la oprirea bruscă a sistemului
- folosește direct API-ul din MTD fără a trece printr-un nivel de translatare (FTL)

Funcționarea JFFS2

- modificarea fișierelor se realizează prin intermediul logurilor
  - logurilor conțin datele ce au fost modificate
  - numai aceste loguri se salvează
  - pentru citirea fișierelor se va citi toate logurile aferente pentru a recrea fișierul
- periodic se rulează un garbage collector pentru a șterge logurile inutile
  - acest garbage collector realizează și wear leveling prin rearanjarea datelor în zonele mai puțin folosite

NFS

- sistemele de fișiere desktop (ext2,3,4...) pot fi exportate și către rețea
- kernelul Linux oferă un mecanism de montare a astfel de sisteme de fișiere prin rețea
- un sistem NFS poate fi montat chiar și ca /, permițând astfel o depanare rapidă a sistemului (prin faptul că aplicațiile pot fi dezvoltate pe un desktop, iar pentru teste acestea nu trebuie înscrise în flash)
- bootloader trebuie să ofere suport pentru conectare la rețea (BOOTP, DHCP)

procfs

- proc este un sistem de fișiere logic, care oferă informații în timp real despre starea kernelului și o interfață prin care se pot modifica anumite variabile interne folosite de kernel
- acest sistem este pur logic, kernelul creând toate fișierele incluse la momentul accesului
  - totuși este nevoie de resurse importante din memorie
- pentru a salva memorie se poate înlătura sistemul proc, dar atunci multe funcții de bază din sistemul Linux nu vor fi accesibile (`ps`, `mount`)

Optimizarea spațiului de stocare

- spațiul de stocare în flash poate fi destul de limitat, astfel e nevoie de optimizarea ocupării acestui spațiu
- kernelul poate fi compilat cu opțiunea -Os pentru a reduce dimensiunea codului
- kernelul 2.6 oferă o opțiune `CONFIG_EMBEDDED`, care permite compilarea unui kernel mai subțire prin posibilitatea înlăturării unor subsisteme esențiale pe desktop, dar neutilizate pe embedded (de ex. swap)

Optimizarea bibliotecilor

- bibliotecile standard poate să consume spațiu inutil cu funcții nefolosite în aplicațiile finale
- în sistemele embedded de obicei aplicațiile finale sunt cunoscute din avans, ce permite o optimizare a bibliotecilor folosite
- se pot utiliza biblioteci reduse, destinate sistemelor embedded: dietlibc, uclibc
- bibliotecile pot fi optimizate cu un tool dedicat, după compilarea tuturor aplicațiilor pentru a elimina funcțiile nefolosite (libraryopt)

Reducerea numărului de aplicații

- sistemele Linux normale includ un număr mare de programe folosite pentru a realiza anumite funcții oferite de sistemul de operare
- aproape toate comenzile ce pot fi emise în linia de comandă sunt programe mici care implementează un apel sistem
- pentru a reduce spațiul folosit toate aceste programe pot fi integrate într-unul singur care încorporează toate apelurile sistem des folosite (de ex. busybox)

Busybox

- este un sistem multicall, care implementează funcțiile oferite de programele Linux:
  - integrează un shell (linie de comandă)
  - programe de bază: `cat`, `cp`, `mv`, `dd`, `rm`, `ls`, `pwd`, ...
  - controlul proceselor: `ps`, `kill`, `...`
  - controlul sistemului: `init`, `reboot`, `...`
  - controlul modulelor: `lsmod`, `rmmod`, `modprobe`, `...`
  - operațiuni cu rețea: `ifconfig`, `route`, `ping`, `tftp`, `telnet`, `wget`, ...
  - unelte de arhivare: `tar`, `gzip`, ...
  - managementul utilizatorilor: `login`, `adduser`, `passwd`, ...
  - ...