IAF0542

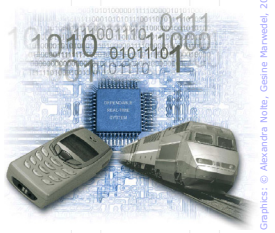
Sardsüsteemid
VIII Loeng

Gert Jervan

Arvutitehnika instituut

ati.ttu.ee/~gerje

www.pld.ttu.ee/IAF0542



Graphics: © Alexandra Nolte, Gesine Marwedel, 2003

Prioriteetidel põhinev katkestav planeerimine

- Priority Based Preemptive Scheduling
  - Programmi ei genereerita ette valmis. Ülesanded käivituvad väliste sündmuste (näiteks signaalide, sõnumite) mõjul
  - Igal ajahetkel jookseb kõrgeima prioriteediga ülesanne. Kui korraga on valmis mitu ülesannet, siis valitakse kõrgeima prioriteediga ülesanne
  - Ülesandeid võib katkestada igal ajahetkel. Kui ülesanne on valmis täitmiseks ning tema prioriteet on kõrgem, kui hetkel täidetaval ülesandel, siis ülesande täitmine katkestatakse

Prioriteetidel põhinev katkestav planeerimine

- Prioriteete võib omistada nii staatiliselt kui ka dünaamiliselt
- Kas ülesanne saab valmis enne oma piir-aega? Sellele leiab vastuse planeeritavuse analüüsiga (schedulability analysis)

Planeeritavuse analüüs

- Planeeritavust on võimalik analüüsida teatud juhtudel:
  - Ülesannete perioodid ja täitmisajad on teada
  - Ülesannete perioodid on staatilised
    või
    Kasutatakse teatud piiratud dünaamilist prioriteetide põhimõtet, nagu näiteks EDF
  - Ülesandeid täidetakse ühel protsessoril
    või
    Multiprotsessorsüsteemid, kus kommunikatsiooniinfrastruktuur on ennustatava viitega (nagu näieks CAN, TDMA protokollid jne.)

The MARS Pathfinder problem (2)

- "VxWorks provides preemptive priority scheduling of threads. Tasks on the Pathfinder spacecraft were executed as threads with priorities that were assigned in the usual manner reflecting the relative urgency of these tasks."
- "Pathfinder contained an "information bus", which you can think of as a shared memory area used for passing information between different components of the spacecraft."
  - A bus management task ran frequently with high priority to move certain kinds of data in and out of the information bus. Access to the bus was synchronized with mutual exclusion locks (mutexes)."

The MARS Pathfinder problem (3)

- The meteorological data gathering task ran as an infrequent, low priority thread, … When publishing its data, it would acquire a mutex, do writes to the bus, and release the mutex. ..
- The spacecraft also contained a communications task that ran with medium priority."

☞

High priority: retrieval of data from shared memory
Medium priority: communications task
Low priority: thread collecting meteorological data

The MARS Pathfinder problem (4)

– "Most of the time this combination worked fine. However, very infrequently it was possible for an interrupt to occur that caused the (medium priority) communications task to be scheduled during the short interval while the (high priority) information bus thread was blocked waiting for the (low priority) meteorological data thread. In this case, the long-running communications task, having higher priority than the meteorological task, would prevent it from running, consequently preventing the blocked information bus task from running. After some time had passed, a watchdog timer would go off, notice that the data bus task had not been executed for some time, conclude that something had gone drastically wrong, and initiate a total system reset. This scenario is a classic case of priority inversion."

Priority inversion on Mars

- Priority inheritance also solved the Mars Pathfinder problem: the VxWorks operating system used in the pathfinder implements a flag for the calls to mutex primitives. This flag allows priority inheritance to be set to "on". When the software was shipped, it was set to "off".

The problem on Mars was corrected by using the debugging facilities of VxWorks to change the flag to "on", while the Pathfinder was already on the Mars [Jones, 1997].

VxWorks – WindRiver RTOS

Sard- ja reaalaja OSid

Gert Jervan

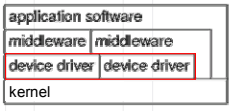
Sard OS - nõudmised

– Konfigureeritavus
Mitte ükski reaalaja OS (RTOS) ei sobi kõikide süsteemide jaoks, meil ei ole võimalust hoida süsteemis kasutamata funktsionaalsust ☞ vajadus kofigureeritavuse järgi.
  - Lihtsam moodus: eemalda mittevajalik funktsionaalsus (linker?).
  - Tingimuslik kompileerimine (kasutade #if ja #ifdef käske).
  - Dünaamilised andmed võidakse asendada staatiliste andmetega.
  - Kompileerimisaegne hindamine.
  - Objektorienteeritus.
– Suurte süsteemide, kus mitmeid erinevaid OSe verifitseerimine võib olla keeruline:
  - Iga tuletatud OS tuleb põhjalikult testida;
  - Näiteks probleem eCos'iga: (open source RTOS from Red Hat) 100 kuni 200 konfigureerimise punkti[Takada, 01].

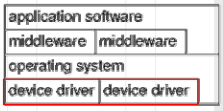
Sard OS – nõudmised (2)

– Ketast ja võrguühendust hallatakse läbi ülesannete, mitte draiverite kaudu.
– Paljud sardsüsteemid on ilma ekraani, klaviatuuri, hiireta.
– Põhimõtteliselt ei ole olemas ühtegi seadet (välja arvatud süsteemi timer), mida peaksid toetama kõiks OSi versioonid.

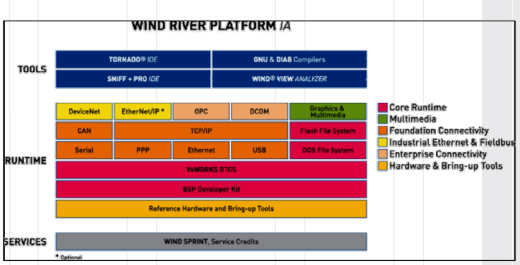
Embedded OS



Standard OS



Näide: WindRiveri platvorm autotööstusele



© Windriver

Sard OS – nõudmised (3)

- Kaitsemehhanismid ei ole alati vajalikud:
  Sardsüsteem on mingi kindla ülesande jaoks, mittetestitud tarkvara ei laadita peaaegu kunagi, tarkvara loetakse usaldusväärseks.
  (NB! Kaitsemehhanisme võib vaja minna ohutuse ja turvalisuse tagamiseks).

Eraldi I/O operatsioone ei ole vaja ning ülesannetel võib olla oma I/O:
Example: Let `switch` be the address of some switch
Simply use

```
load register,switch
```

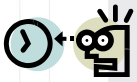
instead of OS call.

Sard OS – nõudmised (4)

- Katkestusi võivad tekitada kõik protsessid
  Standard OSi puhul oleks see suur risk töökindlusle.
  - sardtarkvara kohta võib eeldada, et see on testitud,
  - kuna kaitsmine ei ole oluline ja
  - kuna hea kontroll erinevate seadmete üle on vajalik,
  - on võimalik lasta katkestustel alustada ja peatada ülesandeid (hoides ülesannete algusaadresse katkestuste tabelis).
  - On märgatavalt effektiivsem, kui seda teha läbi OSi.

Sard OS – nõudmised (5)

- Reaalaeg:
  - Paljud sardsüsteemid on reaalajasüsteemid, seetõttu tuleb nendes süsteemides kasutada reaalaja OSe (RTOS).



RTOS

- **Def.:** (A) real-time operating system is an operating system that supports the construction of real-time systems
- The following are the three key requirements:
  - **The timing behavior of the OS must be predictable.**
    ∀ services of the OS: Upper bound on the execution time!
    RTOSs must be deterministic:
    - unlike standard Java,
    - short times during which interrupts are disabled,
    - contiguous files to avoid unpredictable head movements.

[Takada, 2001]

RTOS (2)

- **OS must manage the timing and scheduling**
  - OS possibly has to be aware of task deadlines; (unless scheduling is done off-line).
  - OS must provide precise time services with high resolution.

[Takada, 2001]

Aeg

- Aeg on reaalaja süsteemides kesksel kohal
- Tegelik aeg on reaalsetes numbrites
- Kaks standardit, mida kasutatakse:
  - **International atomic time TAI**
    (french: temps atomic internationale)
    Free of any artificial artifacts.
  - **Universal Time Coordinated (UTC)**
    UTC is defined by astronomical standards
- UTC ja TAI on identsed alates 01.01.1958.
- Sellest ajast alates on lisatud 30 sekundit.

Sisemine sünkroniseerimine

- Sünkroniseeritakse ühe "master clock-iga"
  - Tüüpiliselt algkäivitusel
- Hajutatud sünkroniseerimine:
  - Kogutakse informatsiooni naabritelt
  - Arvutatakse parandusväärtus
  - Muudetakse aega
- Esimese sammu täpsus on sõltuv:
  - Rakenduste tasemel: ~500 µs - 5 ms
  - OSi kernel: 10 µs - 100 µs
  - Kommunikatsiooni riistvara: < 10 µs

Väline sünkroniseerimine

- Väline sünkroniseerimine tagab ühilduvuse globaalse ajaga.
- Viimasel ajal kasutatakse selleks enneköike GPSe
- GPS pakub TAI ja UTC ajainformatsiooni.
- Täpsus on ca 100 ns.

GPS hiir

© Dell

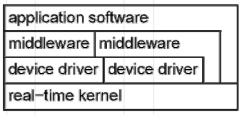
RTOS (3)

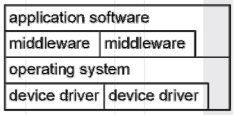
- **OS peab olema kiire!**
  Vajalik praktikas

[Takada, 2001]

RTOSi kernelid

- Eristatakse
  - Reaalaja kerneleid ja standard OSide muudetud kerneleid





✓ **Eristatakse**
- Üldised RTOSid and RTOSid spetsiaalse rakendusvaldkonna jaoks,
- Standardsed APId (e.g. POSIX RT-Extension of Unix, ITRON, OSEK) or spetsiaalsed APId.

RTOS kernelite funktsionaalsus

- Sisaldab
  - Protsessori haldus
  - Mälu haldus
  - Timeri haldus

  } Ressursside haldus
  - Ülesannete haldus (resume, wait etc),
  - Ülesannete vaheline kommunikatsioon ja sünkroniseerimine

Vahevara - middleware

- Reaalaja andmebaasid
- Ligipääs kaugemale olevatele objektidele

Reaalaja andmebaasid

- Eesmärk: hoida ja pakkuda püsivat infot
- Tehing = jada lugemis/kirjutamisoperatsioone
- Muutused ei ole püsivad kuni need ei ole kinnitatud
- Tehingutele esitatavad nõudmised ("ACID"):
  - **A**tomic: state information as if transaction is either completed or had no effect at all.
  - **C**onsistent: Set of values retrieved from several accesses to the data base must be possible in the world modeled.
  - **I**solation: No user should see intermediate states of transactions
  - **D**urability: results of transactions should be persistent.

Reaalaja andmebaasid (2)

- Reaalaja andmebaaside loomisega seotud probleemid:
  - Tehinguid võidakse suvalisel ajahetkel katkestada, ilma et oleks kinnitatud
  - Kõvaketastega töötamine on ääretult ettearvamatu

✓ Võimalikud lahendused
1. Kettavabad andmebaasid
2. Nõrgendatud ACID nõudmised

Reaalaja CORBA

- Väga oluline, et RT-CORBA pakuks
  - Ennustatavust fikseeritud prioriteetidega süsteemis.
    - See sisaldab kliendi ja serveri vahel lõimede prioriteetide austamist, et lahendada ressurssidele konkureerimist
    - ja operatsioonide latentsuse piiramist.
  - Lõimede prioriteete ei ole vaja jälgida, kui lõimed saavutavad välistava (mutually exclusive) ligipääsu ressurssidele (priority inversion).

Message passing interface (MPI)

- Message passing interface (MPI): alternative to CORBA
- MPI/RT: a real-time version of MPI [MPI/RT forum, 2001].
- MPI-RT does not cover issues such as thread creation and termination.
- MPI/RT is conceived as a potential layer between the operating system and standard (non real-time) MPI.

Energia- ja võimsustarve

Miks on energiatarve nii oluline?

- Kaasaskantavad süsteemid – aku eluiga!
- Süsteemid väga limiteeritud energiaeelarvega: Mars Pathfinder, UAV
- Desktopid ja severid: väga suur võimustarve
  - Tõstab temperatuuri ning vähendab jõudlust ning usaldusväärsust
  - Tõstab vajadust kallite jahutusmehhanismide järele
- Üks kõrge jõudlusega kiipide loomise põhitakistusi on kuumuse eemaldamine
- Suur võimsustarve toob kaasa ka majanduslikud ja keskkonna-alased probleemid (green computing)

Võimsustarve CMOS tehnoloogia seadmetes

- CMOS (Complementary Metal-Oxide Semiconductor)

Dünaamiline | Staatiline

$$P = \frac{1}{2} \cdot C \cdot V_{DD}^2 \cdot f \cdot N_{SW} + Q_{SC} \cdot V_{DD} \cdot f \cdot N_{SW} + I_{leak} \cdot V_{DD}$$

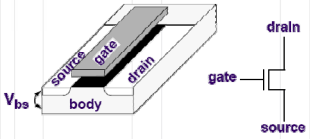
Ümberlülitused Switching Power | Lühised Short circuit power | Lekked Leakage power

C	= node capacitances	V_{DD} =	supply voltage
N_{SW}	= switching activities (number of gate transitions per clock cycle)	Q_{SC} =	charge carried by short circuit current per transition
f	= frequency of operation	I_{leak} =	leakage current

Power, Energy, Voltage, Power consumption
Võimsus, energia, pinge, võimsustarve

Võimsustarve CMOS tehnoloogia seadmetes

CMOS transistor (N-type)



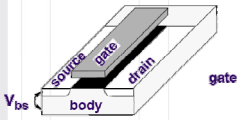
Threshold voltage:
- **The minimal voltage required at the gate to turn on the transistor**

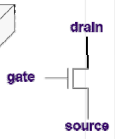
V_{bs} = body bias voltage
V_{th} = threshold voltage

$V_{DD, max}$ = 3,3 V → V_{th} ≈ 0,8 V

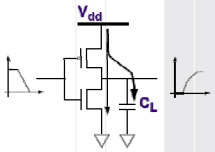
Võimsustarve CMOS tehnoloogia seadmetes

CMOS transistor (N-type)





CMOS inverter

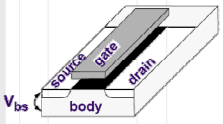


V_{bs} = body bias voltage
V_{th} = threshold voltage
V_{dd} = supply voltage
C_L = output load capacitance

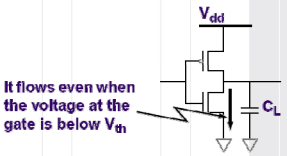
Dynamic power
- **Charging and discharging the output load capacitance**
- **Momentary short circuits at a gate's output**

Võimsustarve CMOS tehnoloogia seadmetes

CMOS transistor (N-type)



CMOS inverter



V_{bs} = body bias voltage
V_{th} = threshold voltage
V_{dd} = supply voltage
C_L = output load capacitance

Static power
- **Subthreshold leakage conduction**
- **Junction leakage (drain and source to body)**

Võimsustarve CMOS tehnoloogia seadmetes

- Pikka aega on lekkevoolu võimsust peetud tühiseks võrreldes dünaamilise võimsusega
- Tänapäeval on need kaks saanud võrreldavateks
- Tehnoloogia arenemisel alla 65 nm hakkab lekkevoolu võimsus ületama dünaamilist

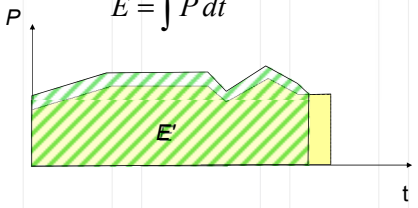
Võimsustarve CMOS tehnoloogia seadmetes

- Lekkevool eksisteerib isegi siis kui seadmed ei ole kasutuses (standby). Ainukene võimalus vabaneda sellest on toiteallika eemaldamine
- Lühiste energia on ca 10% kogu energiast
- Lülituste energia on tänapäevastes kiipides endiselt suurim probleemide allikas
- Edaspidiselt räägime vaid lülituste energiast, kui ei ole mainitud midagi eraldi

Võimsus ja energia

$$P = \frac{1}{2} \cdot C \cdot V_{DD}^2 \cdot f \cdot N_{SW}$$

$$E = \int P\,dt$$



- Paljudel juhtudel tähendab kiirem täitmine ka vähem energiat kuid see võib olla ka vastupidi, kui kiirema täitmise saavutamiseks tuleb võisust tõsta

Võimsustarve v. energiatarve

- Võimsustarbe vähendamine on oluline:
  - Toiteallika disainil
  - Pingeregulaatorite disainil
  - Ühenduste dimensioneerimisel
  - Lühiajalisel jahutamisel
- Energiatarbe vähendamine on oluline:
  - Piiratud energiaressursiga süsteemides (i.e. mobiilsed süsteemid)
    - limiteeritud aku
    - kallis energia
  - Jahutus
    - kõrge hind
    - limiteeritud pind
  - Usaldusväärsus
  - Pikk eluiga, madalad temperatuurid

Võimsus/energiatarbe vähendamine

- Põhilised võimalused:
  - Toitepinge vähendamine
  - Ümberlülituste arvu vähendamine
  - Mahtuvuse vähendamine
  - Tsüklite arvu vähendamine

$$P = \frac{1}{2} \cdot C \cdot V_{DD}^2 \cdot f \cdot N_{SW}$$

$$E = \int P\,dt$$

Võimsus/energiatarbe vähendamine (2)

- Skeemi tasemel
  - Transistoride ümberjärjestamine (mõjutab mahtuvust)
  - Transitoride suurused
- Loogikatasemel
  - "Don't care" (X) optimeeriminem et vähendada ümberlülituste arvu
  - "Valede" ümberlülituste vähendamine läbi viidete ühtlustamise
  - Tehnoloogia sidumine
  - Olekute sobiv kodeerimine, et vähendada ümberlülituste arvu olekumuutusel
  - Kodeerimine, et vähendada ümberlülituste arvu siinil või ALUs
  - Gated clocks

Võimsus/energiatarbe vähendamine (3)

- Käitumuslikul tasemel
  - Planeeri ja seo ülesanded sedasi, et tsüklite arv oleks väiksem (rohkem tegevust ühe takti jooksul) → väiksem töökiirus → madalam toitepinge
  - Hõiva ja jaga mooduleid sedasi, et võimsustarve oleks väiksem

Võimsus/energiatarbe vähendamine (4)

- Arhitektuursel tasemel
  - Spetsiaalne käsustik, andmeosa ja registrite struktuur, mis vastaksid valitud arhitektuurile, eesmärgiga võimsuse vähendamine
    - Kiibil asuvad ja töötavad ainult need ressursid, mida tõesti vaja on
  - Siini võimsustarbe vähendamine
    - Väiksem ümberlülituste arv: tark kodeerimine, aadressisiini lülituste arvu vähendamine kasutades korrelatsioone
    - Siini pikkuse vähendamine ressursside õige paigutamise teel (vähendab mahtuvust)
    - Siini segmentideks jaotamine: pika suure koormusega siini jaotamine kohalikeks segmentideks

Võimsus/energiatarbe vähendamine (5)

- Mälustruktuuri optimeerimine
  - Mälu poole pöördumised on eriti energianäljased. Üks mälusiire võtab 33x rohkem energiat kui liitmisoperatsioon!

    Mälu poole pöördumiste arvu vähendamine on väga edukas meetod võimsustarbe vähendamiseks
  - Cache'i kohandamine (arv, suurus, assotsiatiivsus, rea pikkus) vastavale rakendusele → aitab kokku hoida mälu poole pöördumiste arvu
  - Huvitav küsimus: suuremad cache'id tarbivad rohkem energiat, kuid aitavad vähendada mälu poole pöördumiste arvu. Milline on õige tasakaal?

Võimsus/energiatarbe vähendamine (6)

- Võimsustarbe haldamine (power management):
  - Käsud, mis võimaldavad süsteemi mõningate osade ootele panemist või seiskamist
  - Käsud, mis võimaldavad dünaamiliselt muuta toitepinget

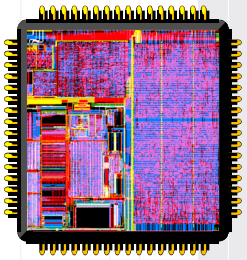
Võimsus/energiatarbe vähendamine (7)

- Süsteemi tasemel
  - Staatilised tehnikad, mida rakendatakse disaini käigus
    - Kompileerimine madala energiatarbe jaoks: käskude valimine, andmete mälusse jaotamine, registrite jaotamine
    - Algoritmi disain: leida algoritm, mis on kõige energiaefektiivsem
    - Ülesannete sidumine ja planeerimine
  - Dünaamilised tehnikad, mida rakendatakse töö käigus
    - Neid tehnikaid rakendatakse töö käigus, et ära kasutada nii ooterežiime, kui ka madala koormuse perioode

ASIC - Application Specific Circuits

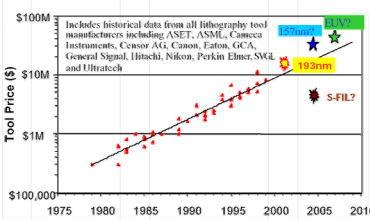
- Spetsiaalskeemid on vajalikud, kui eesmärgiks on:
  - Suurim kiirus
  - Energia efektiivsus

    ja
  - neid saab müüa miljoneid
- Probleemiks
  - Väljatöötamiseks kuluv aeg
  - Paindlikkuse puudus
  - Kõrge hind (maskide hinnad on miljonites dollarites)



Riistvara väljakutsed

- Paindlikkuse puudumine (muutuvad standardid)
- Maskide ülikõrge maksuvus



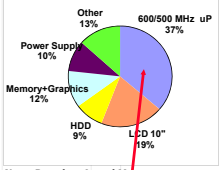
☞ Suund on üha suuremale tarkvara osatähtsusele

[http://www.molecularimprints.com/Technology/tech_articles/MII_COO_NIST_2001.PDF9]

Protsessor v. süsteem

[Courtesy: N. Dutt; Source: V. Tiwari]

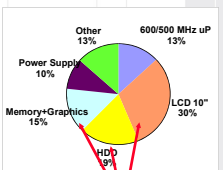
Mobile PC (notebook) Thermal Design (TDP) System Power



Note: Based on Actual Measurements

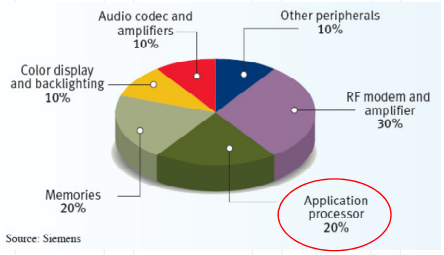
CPU domineerib termovõimsuse osas

Mobile PC (notebook) Average System Power



Mitmed platvormi elemendid on olulised keskmise võimustarbe puhul

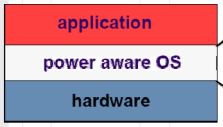
Energiatarve kaasaskantavates seadmetes



[O. Vargas (Infineon Technologies): Minimum power consumption in mobile-phone memory subsystems; Pennwell Portable Design - September 2005;] Thanks to Thorsten Koch (Nokia/ Univ. Dortmund) for providing this source.

Dünaamiline võimuse haldamine (DPM)

DPM: Dynamic Power Management



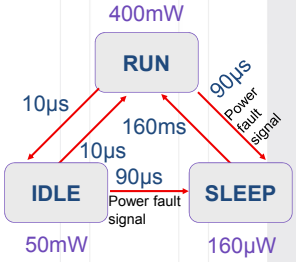
- Otsused:
  - Erinevate olekute vahel ümberlülitamine
    - Idle
    - Sleep
    - Run
  - Erinevate töösageduste ja toitepingete vahel ümberlülitamine

Eesmärgid:
- Energia optimeerimine
- Teenuse kvaliteedi tagamine

Dünaamiline võimuse haldamine (DPM)

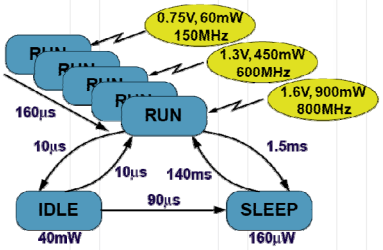
Näide: STRONGARM SA1100

- RUN: tavaline töötamine
- IDLE: tarkvara peatab CPU töö, kuid jälgib katkestusi
- SLEEP: kiibi tegevus peatatakse, äratus läbi "wake-up" sündmuse



Dünaamiline võimuse haldamine (DPM)

- Riistvara toega: Intel Xscale



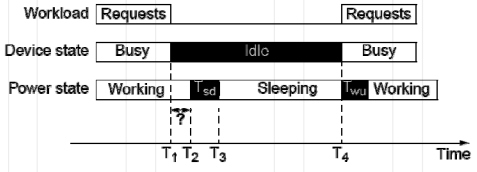
Võimalikud vahepealsed olekud: DEEP IDLE, STANDBY, DEEP SLEEP

Dünaamiline võimuse haldamine (DPM)

- DPMi kasutatakse palju laptopides, PDA'des ja teistes kaasaskantavates seadmetes, et sulgeda või panna ootele mittevajalikke komponente. Eesmärgiks on energia kokkuhoid
- DPMi jaoks on OSide tugi (näiteks Windows 2000 ja uuemad)

DPMi põhimõte

- Kui seadme poole pöördutakse, siis seade on hõivatud, vastasel juhul ootel (idle)
- Kui seade on ootel, siis võib selle kas sulgeda või üle viia madala energiaga ooteasendisse (sleeping)



Dünaamiline pinge muutmine (DVS)

DVS: Dynamic Voltage Scaling

CMOSi energiatarve (lekete ignoreerimisel):

$$P = \frac{1}{2} \cdot C \cdot V_{DD}^2 \cdot f \cdot N_{SW}$$

CMOSi viide:

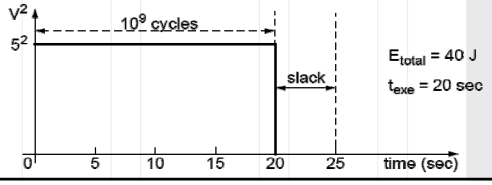
$$\tau = k\,C\,\frac{V_{dd}}{(V_{dd} - V_t)^2} \text{ with}$$

V_t :threshhold voltage ($V_t < \text{than } V_{dd}$)

☞ V_{dd} vähendamisel väheneb P kahekordselt, samas algoritmide täitmiseks kuluv aeg kasvab vaid lineaarselt $E=P \times t$ väheneb lineaarselt (ignoreerides mälusüsteemi ja V_t)

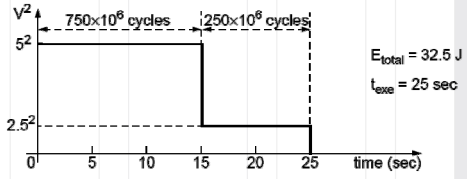
DVS põhimõte

- Olgu meil ülesanne τ:
  - Kogu arvutusaeg on 10^9 tsüklit
  - Deadline: 25 sek
  - Protsessori nominaalne toitepinge: 5V
  - Energia: 40 nJ/tsükkel nominaalsel pingel
  - Protsessori kiirus: 50 MHz (50x10⁶ tsüklit sekundis) nominaalsel pingel



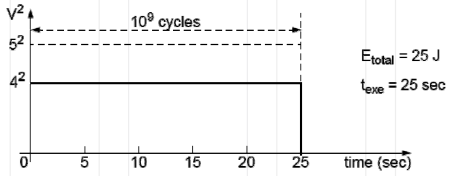
DVS põhimõte (2)

- Teeme aeglasemaks!
  - VDD = 2,5 V
    - Energia: 40x2,5²/5² = 10 nJ/tsükkel
    - Kiirus: 50x2,5/5 = 25 MHz



DVS põhimõte (3)

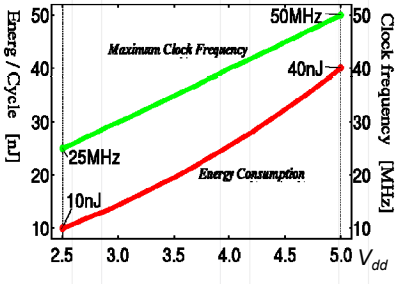
- VDD = 4 V
  - Energia: 40x4²/5² = 25 nJ/tsükkel
  - Kiirus: 50x4/5 = 40 MHz



Dünaamiline pinge muutmine (DVS)

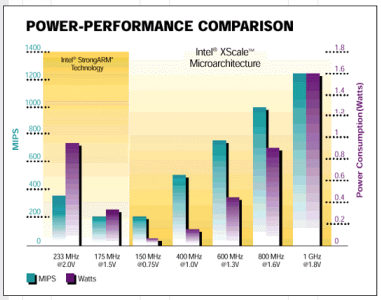
- Transmeta Crusoe protsessor:
  - 32 erinevat pingetaset, vahemikus 1,1 – 1,6 V
  - Taktsignaal vahemikus 200 MHz – 700 MHz (33 MHz sammuga)
  - Siire ühelt pinge/sageduse paarilt teisele võtab ca 20 ms
- Inteli SpeedStep tehnoloogia (Näiteks Mobile Pentium III):
  - 2 pinge/sageduse paari

DVS näide



[Courtesy, Yasuura, 2000]

DVS: Intel Xscale



OS peab tegelema energia-eelarve ajalise jaotusega

www.intel.com

Lekked!

$$P = \underbrace{\frac{1}{2} \cdot C \cdot V_{DD}^2 \cdot f \cdot N_{SW} + Q_{SC} \cdot V_{DD} \cdot f \cdot N_{SW}}_{\text{Dünaamiline}} + \underbrace{I_{leak} \cdot V_{DD}}_{\text{Staatiline}}$$

Dünaamiline: Ümberlülitused Switching Power; Lühised Short circuit power

Staatiline: Lekked Leakage power

Dünaamiline väheneb V_{DD} vähenemisel, ajast sõltumata

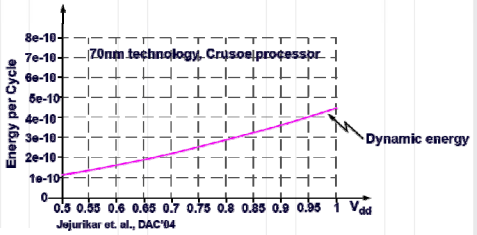
Lekked vähenevad V_{DD} vähenemisel, kuid kasvavad koos ajaga

- Me oleme siiani rääkinud mitte globaalsest energia vähendamisest vaid ainult selle ühe osa vähendamisest
- Kuid selle tulemusena võib energiatarve koguni hoopis suureneda!

Lekked! (2)

$$P = \underbrace{\frac{1}{2} \cdot C \cdot V_{DD}^2 \cdot f \cdot N_{SW} + Q_{SC} \cdot V_{DD} \cdot f \cdot N_{SW}}_{\text{Dünaamiline}} + \underbrace{I_{leak} \cdot V_{DD}}_{\text{Staatiline}}$$

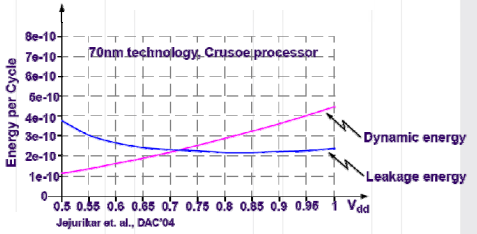
Ümberlülitused Switching Power; Lühised Short circuit power; Lekked Leakage power



Lekked! (2)

$$P = \underbrace{\frac{1}{2} \cdot C \cdot V_{DD}^2 \cdot f \cdot N_{SW} + Q_{SC} \cdot V_{DD} \cdot f \cdot N_{SW}}_{\text{Dünaamiline}} + \underbrace{I_{leak} \cdot V_{DD}}_{\text{Staatiline}}$$

Ümberlülitused Switching Power; Lühised Short circuit power; Lekked Leakage power

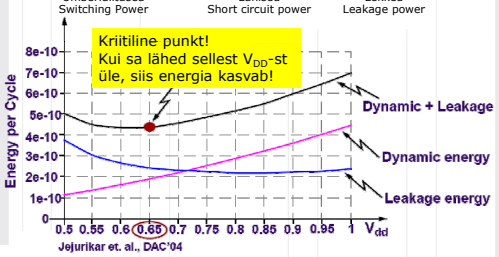


Lekked! (2)

$$P = \underbrace{\frac{1}{2} \cdot C \cdot V_{DD}^2 \cdot f \cdot N_{SW} + Q_{SC} \cdot V_{DD} \cdot f \cdot N_{SW}}_{\text{Dünaamiline}} + \underbrace{I_{leak} \cdot V_{DD}}_{\text{Staatiline}}$$

Ümberlülitused Switching Power; Lühised Short circuit power; Lekked Leakage power

Kriitiline punkt!
Kui sa lähed sellest V_{DD}-st üle, siis energia kasvab!



Küsimusi?

Gert Jervan
ati.ttu.ee/~gerje