TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Arvutitehnika instituut
ati.ttu.ee

Sardsüsteemid
(Embedded Systems)

III Loeng

Gert Jervan
Arvutitehnika instituut

www.pld.ttu.ee/~gerje

Graphics: © Alexandra Nolte, Gesine Marwedel, 2003

Osa materjale: Petru Eles, Peter Marwedel

Ülevaade

- ✓ Eelmise loengu kokkuvõte
- ✓ Sardüsteemid ja usaldusväärsus
- ✓ Arhitektuurid
- ✓ Võimsus- ja energiatarve
- ✓ Protsessorid
- ✓ Platvormid

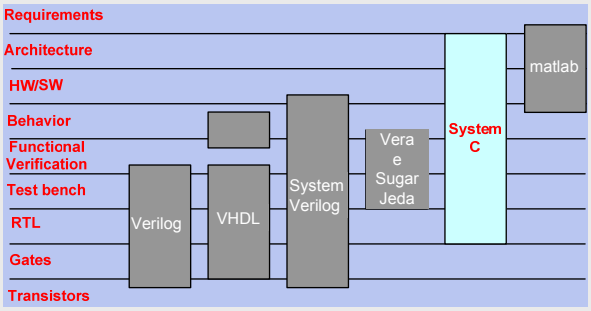
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Arvutitehnika instituut
ati.ttu.ee

Eelmise loengu kokkuvõte

Tallinna Tehnikaülikool
Arvutitehnika instituut

Keeled ja abstraktsioonitasemed



Keelte võrdlus

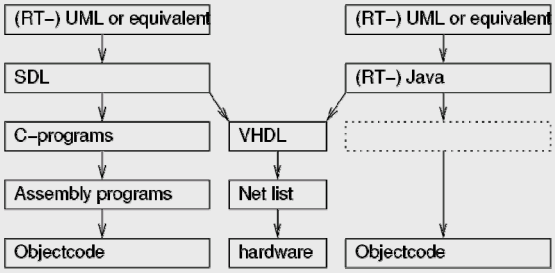
Communication/ local computations	Shared memory	Message passing Synchronous | Asynchronous	
Communicating finite state machines	StateCharts		SDL
Data flow model	Not useful		Kahn process networks
Von Neumann model	C, C++, Java	C, C++, Java with libraries CSP, ADA |	
Discrete event (DE) model	VHDL, Verilog, SystemC	Only experimental systems, e.g. distributed DE in Ptolemy	

Keelde võrdlus

Language	Behavioral Hierarchy	Structural Hierarchy	Programming Language Elements	Exceptions Supported	Dynamic Process Creation
StateCharts	+	-	-	+	-
VHDL	+	+	+	-	-
SpecCharts	+	-	+	+	-
SDL	+-	+-	+-	-	+
Petri nets	-	-	-	-	+
Java	+	-	+	+	+
SpecC	+	+	+	+	+
SystemC	+	+	+	- (2.0)	- (2.0)
ADA	+	-	+	+	+

Keelte kasutamine praktikas

Erinevad lähenemised:



Milline modelleerimissuund valida?

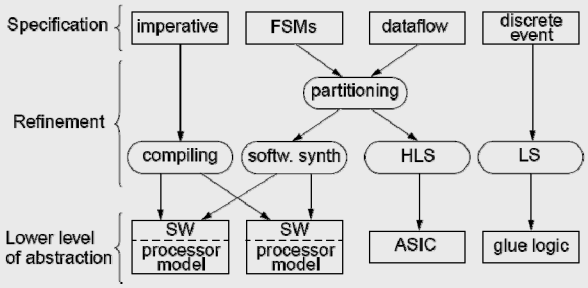
- Kõik sõltub loodavast süsteemist
  - Kas domineerib andmevoog (nagu näiteks DSPdes) või on tegemist kontrollile orienteeritud süsteemidga (reaktiivsed süsteemid)?
  - Sünkroonne või asünkroonne? Tsentraliseeritud või hajutatud?
  - Kui suur?
  - Milline on suhe aega?
  - ...
- Mida sa tahad mudeliga teha?
  - Simuleerimine
  - Formaalne verifitseerimine
  - Automaatne süntees
  - ...
- Millised tööriistad on kättesaadavad ja mis sulle (või su bossile) sobivad

Milline modelleerimissuund valida? (2)

- Ära kasuta seda, mis võimaldab teha "kõike"!
- Eelmisest loengust:
  - Suur väljendusvõime: imperatiivne mudel (näiteks C või Java piiranguteta kasutamine):
    - Võime kirjeldada "kõike"
    - Puudub võimalus formaalseks analüüsiks (või see on väga keerukas)
  - Piiratud väljendusvõime, mis põhineb hästi valitud arvutusmudelil:
    - Spetsifitseerida saab ainult valitud süsteeme
    - Formaalne analüüs on võimalik
    - Efektiivse (võib-olla isegi automaatse) sünteesi võimalus

Milline modelleerimissuund valida? (3)

- Suured sardsüsteemid on heterogeensed → mudelite segu



Keelte kasutamine

- Keele valik on suuresti seotud kasutatava modelleerimismeetodiga
  Seda seetõttu, et paljud keeled on väga tugevalt seotud mingi kindla arvutusmudeliga
  - Kommunikeeruvad asünkroonsed olekuautomaadid: SDL, Lotos
  - Sünkroonsed süsteemid: Esterel, StateCharts
  - Andmevoog ja pidevad arvutused: Matlab, Lustre, Silage

Keelte kasutamine (2)

- Osad keeled aga ei põhine ühelgil kindlal arvutusmudelil
  - Tavalised programmeerimiskeeled
    - Imperatiivsed: C, C++, Java, Ada
    - Funktsionaalsed: Lisp, Scheme, Haskell
    - Loogika: Prolog
  - Riistvara kirjelduskeeled
    - VHDL, Verilog, SystemC: imperatiivsed, diskreetsed sündmused
- Kui spetsifikatsioonide kirjeldamiseks kasutatakse teatud piiranguid ning suuniseid, siis nendes keeltes saab kirjeldada enamuste arvutsumudelite põhiseid spetsifikatsioone

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Arvutitehnika instituut
ati.ttu.ee

Sardsüsteemid ja usaldusväärsus

Tallinna Tehnikaülikool
Arvutitehnika instituut

Nõudmised usaldusväärsusele

- ✓ On süsteeme, kus lubatakse vaid 1 rike 10^9 töötunni kohta, see tähendab 1 rike 114 000 aasta kohta
  - ~ 1000 korda väiksem tüüpilisest kiipide rikete arvust.
  - Ohutuskriitilistes süsteemides peavad süsteemid olema töökindlamad, kui selle komponendid individuaalselt.
  - Tuleb kasutada veakindlust suurendavaid mehhanisme.
- ✓ Madal lubatud rikete arv → süsteemid ei saa olla 100% testitavad.
  - Ohutus on kombinatsioon testimisest ja analüüsist. Liskas peab arvestama nii disaini käigus tehtud vigadega kui ka inimeste poolt põhjustatud riketega.

Kopetz'i 12 disainipõhimõtet (1-3)

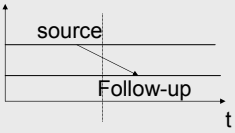
1. Safety considerations may have to be used as the important part of the specification, driving the entire design process.
2. Precise specifications of design hypotheses must be made right at the beginning. These include expected failures and their probability.
3. Fault containment regions (FCRs) must be considered. Faults in one FCR should not affect other FCRs.

Passenger compart-ment stable

Safety-critical & non-safety critical electronics

Kopetz'i 12 disainipõhimõtet (4-6)

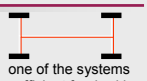
4. A consistent notion of time and state must be established. Otherwise, it will be impossible to differentiate between original and follow-up errors.
5. Well-defined interfaces have to hide the internals of components.
6. It must be ensured that components fail independently.



2 independent brake hose systems

Kopetz'i 12 disainipõhimõtet (7-9)

7. Components should consider themselves to be correct unless two or more other components pretend the contrary to be true (principle of self-confidence).
8. Fault tolerance mechanisms must be designed such that they do not create any additional difficulty in explaining the behavior of the system. Fault tolerance mechanisms should be decoupled from the regular function.
9. The system must be designed for diagnosis. For example, it has to be possible to identifying existing (but masked) errors.



one of the systems sufficient for braking

Kopetz'i 12 disainipõhimõtet (10)

10. The man-machine interface must be intuitive and forgiving. Safety should be maintained despite mistakes made by humans

airbag

Kopetz'i 12 disainipõhimõtet (11-12)

11. Every anomaly should be recorded. These anomalies may be unobservable at the regular interface level. Recording to involve internal effects, otherwise they may be masked by fault-tolerance mechanisms.

12. Provide a never-give up strategy. ES may have to provide uninterrupted service. Going offline is unacceptable.

IAF0030
Arvutitehnika erikursus I

Veakindlad arvutisüsteemid
Fault Tolerant Computer Systems

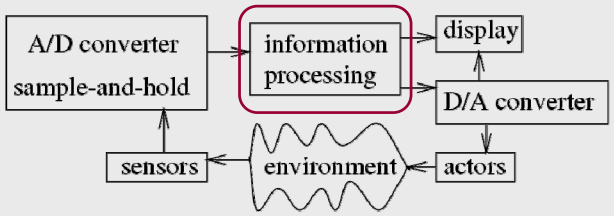
1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY
Arvutitehnika instituut
ati.ttu.ee

Arhitektuurid ja platvormid

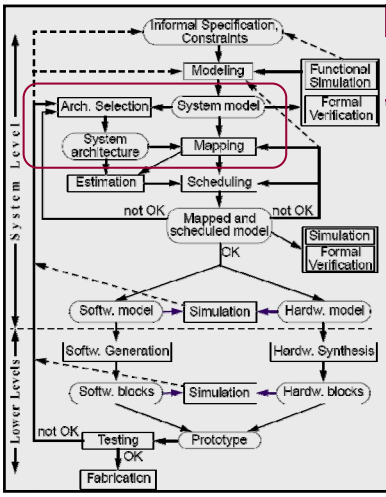
Tallinna Tehnikaülikool
Arvutitehnika instituut

Sardsüsteemide riistvara

- Sardsüsteemide riistvara kasutatakse tihti tsüklis:

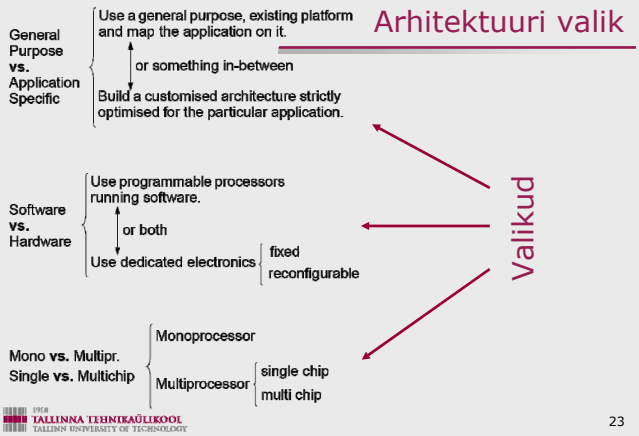


Disainivoog



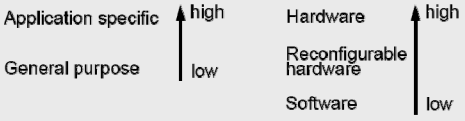
- Arhitektuuri valimine ja sidumine
  - Funktsionaalsus tuleb siduda süsteemi komponentidega
  - Sisaldab nii ülesandeid kui ka kommunikatsiooni

Arhitektuuri valik

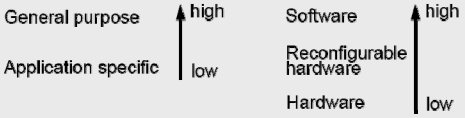


Arhitektuuri valik (2)

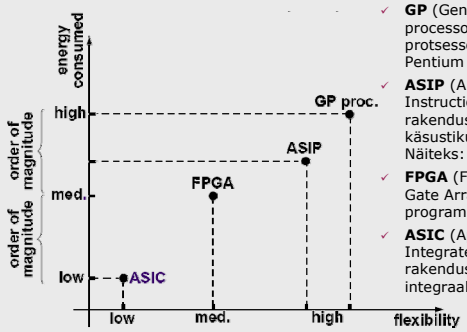
- Põhilised kompromissid:
  - Performance (high speed, low power consumption)



  - Flexibility (how easy it is to upgrade or modify)



Arhitektuuri valik (3)



- **GP** (General Purpose processor) – Tavaline laiatarbe protsessor, i.e. x86 perekond, Pentium jms.
- **ASIP** (Application Specific Instruction set Processors) – rakendus-spetsiifilise käsustikuga protsessor. Näiteks: i960
- **FPGA** (Field Programmable Gate Array) – programmeeritav loogika
- **ASIC** (Application Specific Integrated Circuit) – rakendusspetsiifiline integraalskeem

TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY
Arvutitehnika instituut ati.ttu.ee

Energia- ja võimsustarve

Tallinna Tehnikaülikool
Arvutitehnika instituut

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

Võimsustarve CMOS tehnoloogia seadmetes

- CMOS (Complementary Metal-Oxide Semiconductor)

Dünaamiline Staatiline

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

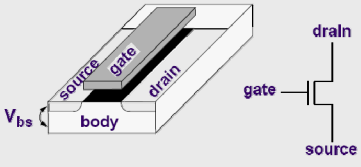
Ümberlülitused Switching Power | Lühised Short circuit power | Lekked Leakage power

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 (2)

CMOS transistor (N-type)



Threshold voltage:

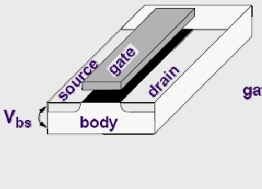
- The minimal voltage required at the gate to turn on the transistor

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

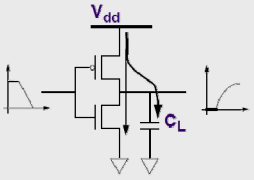
$V_{DD, max}$ = 3,3 V → $V_{th} \approx$ 0,8 V

Võimsustarve CMOS tehnoloogia seadmetes (3)

CMOS transistor (N-type)



CMOS inverter



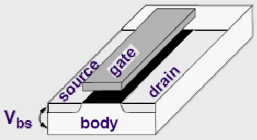
V_{bs} = body bias voltage
V_{th} = threshold voltage
V_{dd} = supply voltage
C_L = output load capacitance

Dynamic power

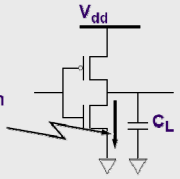
- Charging and discharging the output load capacitance
- Momentary short circuits at a gate's output

Võimsustarve CMOS tehnoloogia seadmetes (4)

CMOS transistor (N-type)



CMOS inverter



It flows even when the voltage at the gate is below V_{th}

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 (5)

- ✓ 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

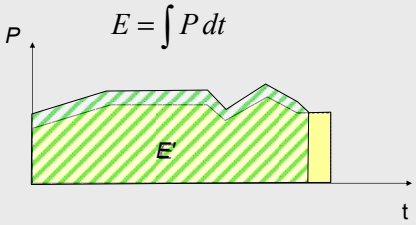
Võimsustarve CMOS tehnoloogia seadmetes (5)

- ✓ 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

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

Võimsus ja energia

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



- ✓ 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)
  - Transistoride 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)

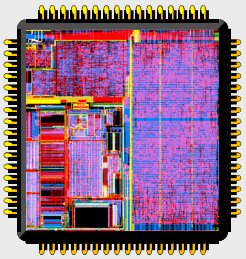
- Süsteemi tasemel
  - Staatilised tehnikad, mida rakendatakse disaini käigus
    - Kompileerimine madala energiatarbe jaoks: käskude valimine, andmete mälusse jaotamine, registrite jaotamine
    - Algoritmi disain: leida algoritm, mis on kõige energiaefektiivsem
    - Ülesannete sidumine ja planeerimine
  - 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

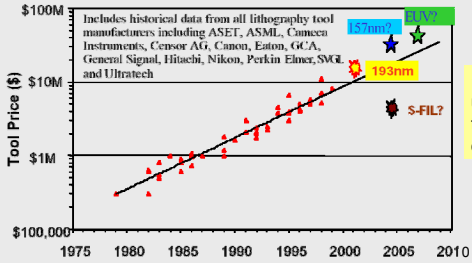
  ja

  - 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



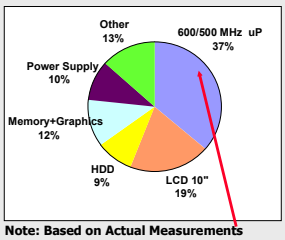
☞ Suund on üha suuremale tarkvara osatähtsusele

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

Protsessor v. süsteem

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

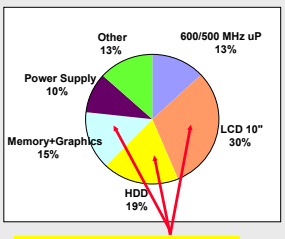
Mobile PC (notebook) Thermal Design (TDP) System Power



Note: Based on Actual Measurements

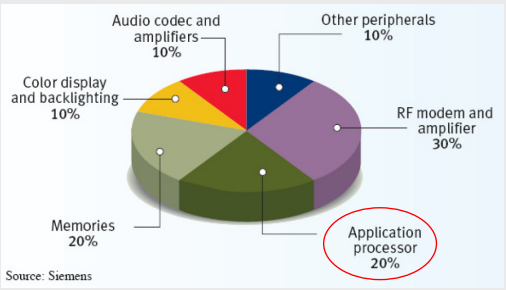
CPU domineerib termovõimsuse osas

Mobile PC (notebook) Average System Power



Mitmed platvormi elemendid on olulised keskmise võimustarbe puhul

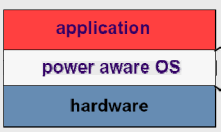
Energiatarve kaasaskantavates seadmetes



[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



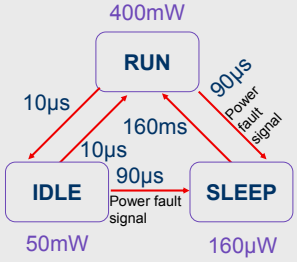
- Otsused:
  - Erinevate olekute vahel ümberlülitamine
    - Idle
    - Sleep
    - Run
  - Erinevate töösageduste ja toitepingete vahel ümberlülitamine

Eesmärgid:
- Energia optimeerimine
- Teenuse kvaliteedi tagamine

Dünaamiline võimuse haldamine (DPM)

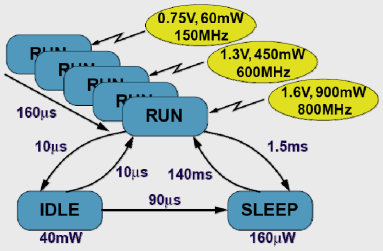
Näide: STRONGARM SA1100

- 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



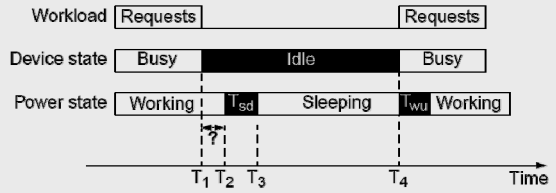
Võimalikud vahepealsed olekud: DEEP IDLE, STANDBY, DEEP SLEEP

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

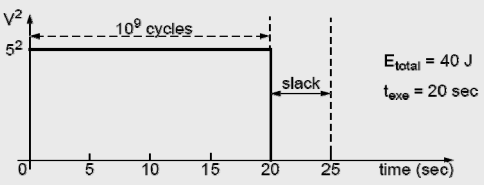
($V_t < \text{than } V_{dd}$)

☞ 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)

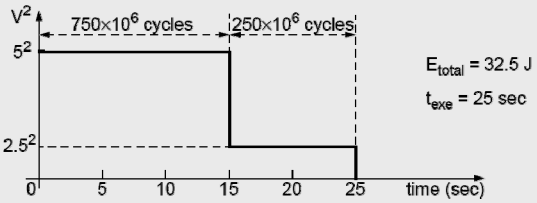
DVS põhimõte

- 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 (50x10^6 tsüklit sekundis) nominaalsel pingel



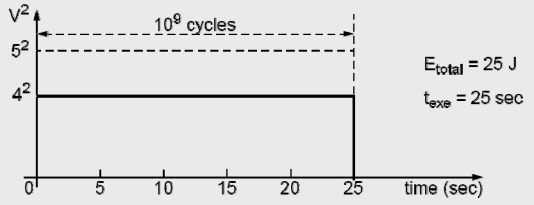
DVS põhimõte (2)

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



DVS põhimõte (3)

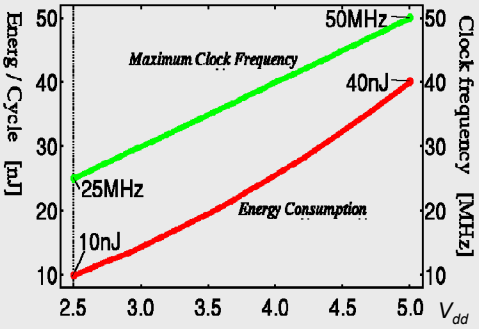
- ✓ VDD = 4 V
  - Energia: $40x4^2/5^2$ = 25 nJ/tsükkel
  - Kiirus: 50x4/5 = 40 MHz



Dünaamiline pinge muutmine (DVS)

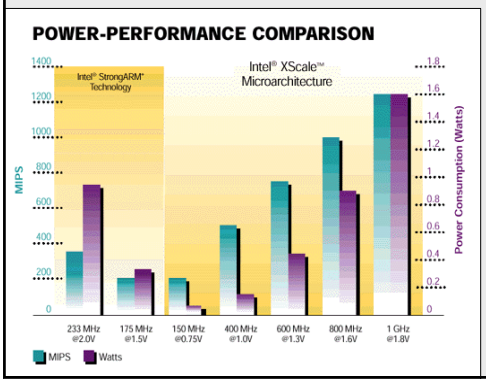
- ✓ Transmeta Crusoe protsessor:
  - 32 erinevat pingetaset, vahemikus 1,1 – 1,6 V
  - Taktsignaal vahemikus 200 MHz – 700 MHz (33 MHz sammuga)
  - Siire ühelt pinge/sageduse paarilt teisele võtab ca 20 ms
- ✓ Inteli SpeedStep tehnoloogia (Mobile Pentium III):
  - 2 pinge/sageduse paari

DVS näide



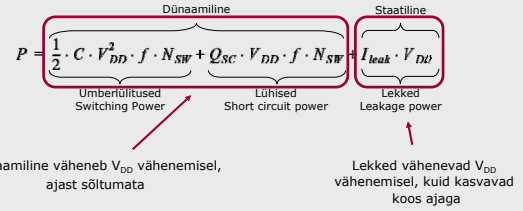
[Courtesy, Yasuura, 2000]

DVS: Intel Xscale



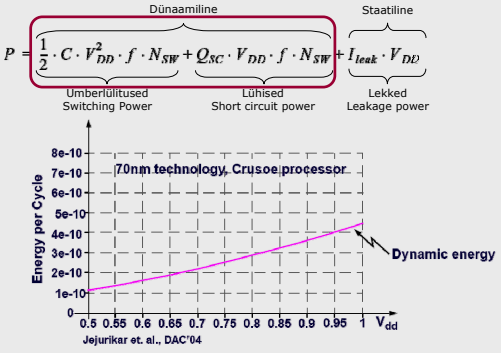
OS peab tegelema energia-eelarve ajalise jaotusega

Lekked!



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

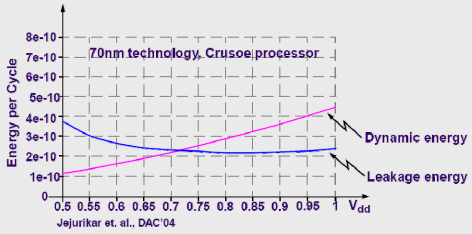
Lekked! (2)



Lekked! (2)

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

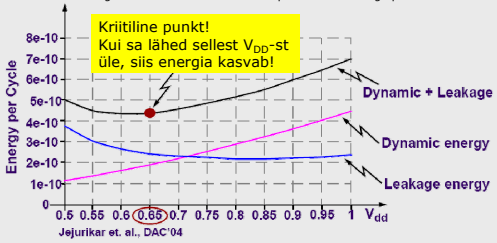
Ümberlülitused Switching Power | Lühised Short circuit power | Lekked Leakage power



Lekked! (2)

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

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



1918 TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

Arvutitehnika instituut
ati.ttu.ee

Protsessorid

Tallinna Tehnikaülikool
Arvutitehnika instituut

Protsessorid

- ✓ Laiatarbe protsessorid ja ASIPid võivad olla nii RISCid, CISCid, DSPd, mikrokontrollerid jms.
  - DSPd ja mikrokontrollerid võivad olla spetsiaalselt loodud DSPks ja kontrolliks
  - Kuid rakendus-spetsiifiline DSP või mikrokontroller on palju spetsiifilisemad
- ✓ Laiatarbe protsessorid:
  - Ei käsustik, mikroarhitektuur ega mälusüsteem ei ole kohandatud ühegi rakenduse või valdkonna jaoks
- ✓ ASIPid
  - Käsustik, mikroarhitektuur ja/või mälusüsteem on kohandatud spetsiaalse rakenduse või valdkonna jaoks
  - Selle tulemusel on saavutatav suurem jõudlus ning väiksem võimsustarve

Mille poolest on ASIP "spetsiifiline"

- ✓ Mille poolest annab protsessorit teha "spetsiifiliseks"
  - Käsustiku (Instruction Set) spetsialiseerimine
    - Eemaldada käsud, mida ei ole vaja
      - vähendab käsusõna pikkust (vähem bitte kodeerimiseks)
      - hoiab kontrolleri ja andmeosa lihtsana
    - Kasutusele võtta antud rakendusele vajalikud spetsiifilised käsud. Isegi eksootilised: aritmeetiliste operatsioonide kombinatsioonid (multply-accumulate), väikesed algoritmid (kodeerimine/dekodeerimine, filtrid), vektoroperatsioonid, stringide töötlemine, pikslite töötlus jne.
      - Vähendab koodi suurust → vähendab mälu hulka, mälu lugemise laiust, võimsustarvet, täitmise aega

Mille poolest on ASIP "spetsiifiline" (2)

- ✓ Funktsionaalsete blokkide ja andmeosade spetsialiseerimine:
  - Kui spetsiifiline käsustik on defineeritud, siis selle implementeerimimiseks on võimalik kasutada spetsiaalset andmeosa ja rohkem või vähem spetsiifilisi funktsionaalseid blokke
    - Sõna pikkuse kohandamine
    - Registrite arvu kohandamine
    - Funktsionaalsete blokkide kohandamine:
      - Võib kasutusele võtta väga spetsiifilisi funktsionaalseid blokke stringide töötlemiseks, pikslite töötluseks, aritmeetikaks ja isegi keerulisi blokke mingite käsujadade täitmiseks (kaasprotsessorid)

Mille poolest on ASIP "spetsiifiline" (3)

- Spetsiifiline mälu
  - Mälublokkide arv ja suurus
  - Mälu ligipääs
    - Mõlemad mõjutavad mälu poole pöördumise paralleelsust
    - Mitu väikest blokki (ühe suure asemel) suurendavad paralleelsust ja kiirust ning vähendavad võimsustarvet
    - Keerulised mälustruktuurid võivad suurendada maksumust ja nõudmisi andmesiinile
  - Vahemälu (cache) konfiguratsioon
    - eraldi andmed/instruktsioonid
    - assotsiatiivsus
    - suurus
    - ridade suurus

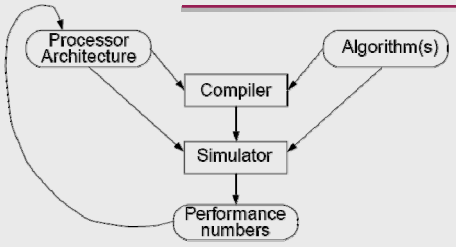
Sõltub väga palju antud rakenduse iseloomust ning enneköike lokaalsusest.

Väga suur mõju jõudlusele ja võimsutarvele

Mille poolest on ASIP "spetsiifiline" (4)

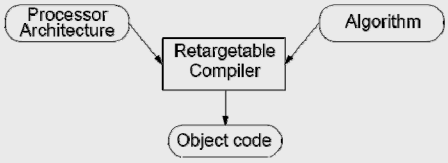
- Ühenduste spetsialiseerimine
  - Funktsionaalsete moodulite ja registrite ühendused
  - Mälu ja cache'i ühendused
    - Kui mitu sisemist siini?
    - Milline protokoll?
    - Täiendavad ühendused annavad täiendavaid võimalusi parallelismiks
- Kontrolli spetsialiseerimine
  - Tsentraliseeritud või hajutatud kontroll (globaalselt asünkroonne)?
  - Konveierid?
  - "Out of order" täitmine?
  - Aparatuurne või mikroprogrammeritud?

ASIPite disainivoog



- Et saada spetsiaalset arhitektuuri on vaja:
  - Ümber häälestatavat kompilaatorit (retargetable)
  - Konfigureeritavat simulaatorit

Retargetable compiler



Retargetable compiler (2)

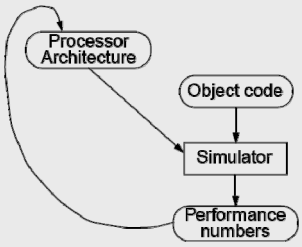
- Automaatselt ümber häälestatavat kompilaatorit on võimalik kasutada erinevate arhitektuuride jaoks

  Koodi optimeerimine ja koodi genereerimine teostatakse kompilaatori poolt ja põhineb arhitektuuri kirjeldusel. Arhitektuuri kirjeldus on antud n.ö. arhitektuuri kirjelduse keeles

- Hea kompilaator ei ole vajalik mitte ainult protsessori spetsialiseerimise käigus

  Sobiva ASIPi puhul on vaja head kompilaatorit ka selle efektiivseks kasutamiseks

Konfigureeritav simulaator

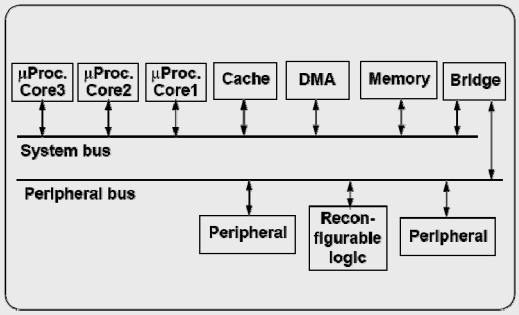


- Sellist simulaatorit on võimalik häälestada sobiva arhitektuuri jaoks (põhinedes arhitektuuri kirjeldusele)
- Kõige olulisem väljund on jõudust kajastavad numbrid:
  - Läbilaskevõime
  - Viide
  - Energia/võimsustarve

Spetsialiseeritud platvormid

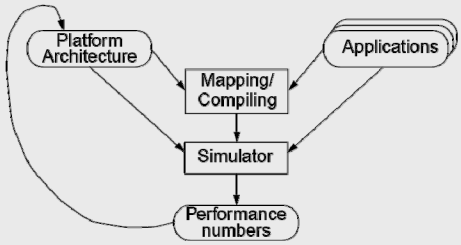
- Mitte ainult protsessoreid vaid ka riistvara platvorme on võimalik spetsialiseerida teatud rakenduste klassi jaoks
- Platvorm defineerib kommunikatsiooni infrastruktuuri (siinid ja protokollid), protsessorite tuumade tüübid, perifeeria, kiirendid ja põhilise mälustruktuuri

Spetsialiseeritud platvormid (2)



Spetsialiseeritud platvormid (3)

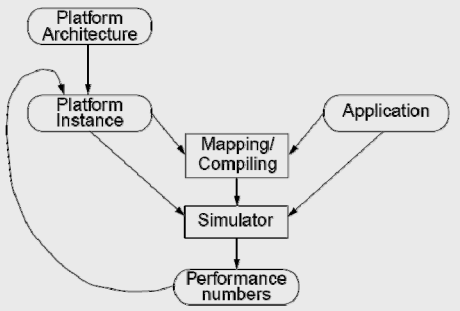
- Lahenduste ruumi uurimine platvormi defineerimisel



Platvormi kohandamine

- Vastava rakenduse jaoks ei looda uut kiipi (mis koosneks sõltumatutest blokkidest) vaid selle jaoks kohandatakse valitud platvormi (instantiation).
- Mida tähendab platvormi kohandamine:
  - Mälu ja cache'i suuruste valik
  - Tuumade ja perifeeria valik
  - Täiendavate ASICute ja kiirendite lisamine
  - Väliprogrammeeritava loogika lisamine

Platvormi kohandamine



Süsteemi platvormid

- Eelnevalt sai räägitud riistvara platvormidest
- Kuid riistvara antakse tavaliselt üle koos tarkvara kihiga:
  riistvara platvorm + tarkvara kiht = süsteemi platvorm
  - Tarkvara kiht:
    - Real-time OS
    - Draiverid
    - Võrguprotokolli stack
    - Kompilaatorid
  - Tarkvara kiht on vahekiht riistvara ja rakendustarkvara vahel (riistvara abstraktsioon)

IP-põhine disain

- Peamine põhimõte tootlikuse tõstmiseks: taaskasutamine (reuse)

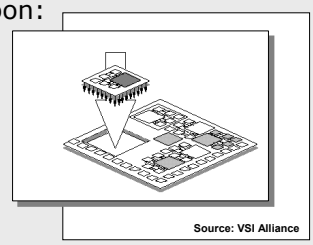
  Et luua uut süsteemi ei ole mõtet alustada nullist vaid tuleks kasutada nii palju kui võimalik (kas siis eelnevatest disainidest pärit või turul saada olevaid) IP blokke (tuumasid)

  IP: Intellectual property; Core: tuum

- Seda kutsutakse: IP-based design, core-based design, reuse techniques jne.
  Core-based design: süsteemi ehitamine olemasolevate komponentide kokku panemise teel

Kiipsüsteemid - Systems-on-Chip

- Milliseid blokke kasutatakse
  - Liidesed, kodeerijad/dekoodeerijad, filtrid, mälud, mikrokontrollerid, DSPd, RISCid, GPd
- Üks võimalik definitsioon:
  - Tuum on blokk, mis on suurem, kui tüüpiline RTL komponent
- Ka tarkvara taaskasutamine



Source: VSI Alliance

Tuumad

- Eelnevalt loodud ja verifitseeritud blokid
  - Pehmed tuumad (Soft cores)
    - Sünteesitav HDL (RTL või kõrgem)
  - Firm cores
    - Teegi komponentide netlist (optimeerituna valitud tehnoloogiasse)
  - Kõvad tuumad (Hard cores)
    - Layoudi kujul, optimeerituna. Ainult funktsionaalne spec on saadaval.

Tuumad (2)

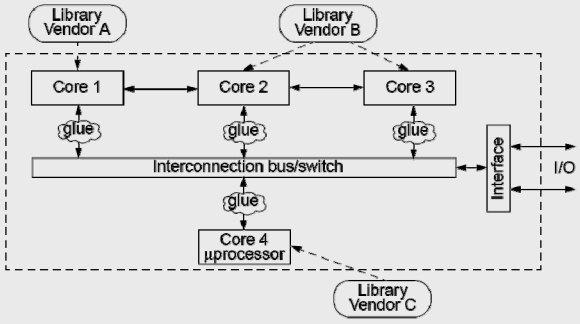
- Hard cores: are fully designed, placed, and routed by the supplier.
  - ⇩ A completely validated layout with definite timing
    - ⇩ rapid integration
    - ⇩ low flexibility
- Firm cores: technology-mapped gate-level netlists.
  - ⇩ less predictability
  - ⇩ flexibility during place and route

Tuumad (3)

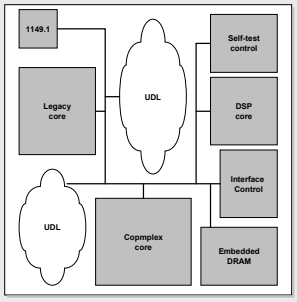
- Soft cores: synthesizable RTL or behavioral descriptions.
  - ⇩ much work with integration and verification.
  - ⇩ maximal flexibility

Flexibility can provide opportunities like e.g. adding application specific instructions to a processor core by modifying the behavioral description.

IP-põhine disain

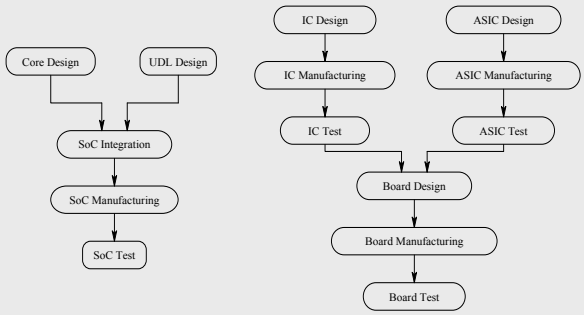


SoC: Arhitektuud



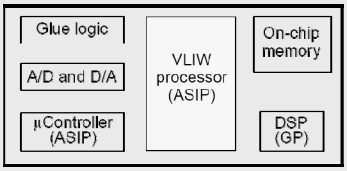
- ✓ 10% UDL
- ✓ 75% mälu
- ✓ 50% "oma" tuumasid
- ✓ 60-70% pehmeid tuumasid

SoC v. SoB



SoC Process | System-on-Board Process

Kiipsüsteem (SoC) multimeedia jaoks



- ✓ Tüüpiline rakendus-spetsiifiline platvorm. Loodud ühte tüüpi rakenduste valdkonnale
- ✓ Peale GP protsessorite tuumade sisaldab ka ASIPi tuumasid, mis on spetsialiseeritud antud rakenduste valdkonna jaoks
- ✓ Rakendus-spetsiifiline mikrokontroller teostab süsteemi üldist kontrolli ning mälu ligipääsu kontrolli
- ✓ Standartne (GP) DSP teostab arvutuslikult vähem nõudlikke modemi ja heli kodek operatsioone
- ✓ VLIW ASIP teostab arvutuslikult nõudlikke operatsioone: diskreetne ja tagurpidi diskreetne koosinus teisendusi, liikumise arvutamist jms.

Kokkuvõte

- ✓ Eelmise loengu kokkuvõte
- ✓ Sardüsteemid ja usaldusväärsus
- ✓ Arhitektuurid
- ✓ Võimsus- ja energiatarve
- ✓ Protsessorid
- ✓ Platvormid

TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

Arvutitehnika instituut ati.ttu.ee

Küsimusi?

Gert Jervan
www.pld.ttu.ee/~gerje

Tallinna Tehnikaülikool
Arvutitehnika instituut