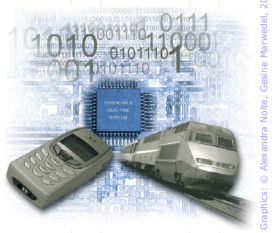
IAF0542

Sardsüsteemid
IX Loeng

Gert Jervan
Arvutitehnika instituut

ati.ttu.ee/~gerje

www.pld.ttu.ee/IAF0542



Graphics: © Alexandra Nolte, Gesine Marwedel, 2003

Sardsüsteemid ja usaldusväärsus

Nõudmised usaldusväärsusele

- **Sardsüsteemid peavad olema usaldusväärsed (dependable)**
- 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. Lisaks peab arvestama nii disaini käigus tehtud vigadega kui ka inimeste poolt põhjustatud riketega.

Veakindlus

- Veakindlaks nimetatakse süsteeme, mis jätkavad oma ettenähtud ülesannete täitmist isegi siis, kui esinevd rikked:
  - riistvaras
  - tarkvaras
  - kasutaja eksimused
  - keskkond, sisendandmed, ...
- Veakindlus on eeldus, mis võimaldab süsteemil saavutada veakindla opereerimise

Põhikontseptsioon

- **Veakindlus (fault tolerance)** on väga tihedalt seotud **usaldusväärsusega (dependable)**,
  - **Töökindlus (Reliability) *R(t)* =** tõenäosus, et süsteem töötab korralikult kui ta töötas ajahetkel *t*=0
  - **Remonditavus (Maintainability) *M(d)* =** tõenäosus, et süsteem töötab taas korralikult d ajaühikut peale vea esinemist.
  - **Töövalmidus (Availability *A*(*t*))**: Tõenäosus, et süsteem töötab ajahetkel *t*
  - **Ohutus (Safety)**: Süsteem ei põhjusta kahju.
  - **Turvalisus (Security)**: Konfidentsiaalne ja usaldatav kommunikatsioon

Isegi ideaalselt loodud süsteemid võivad läbi kukkuda, kui eeldused süsteemi töökoormuse võimalike vigade kohta on valed.
Süsteeme ei saa teha usaldusväärseks tagantjärgi, vaid sellega tuleb arvestada süsteemi loomise algusest alates.

Põhiterminoloogia

- Viga (fault): süsteemis esinev defekt või situatsioon, mis võib viia süsteemi töö tõrkeni
- Rike (error): vea avaldumine – vale käitumine
- Tõrge (failure): süsteem ei täida oma ettenähtud funktsiooni

Viga → Rike → Tõrge

Vigade näited

- Bit-flipid seoses kosmilise kiirgusega:
  - A person on an airplane over the Atlantic at 35,000 ft working on a laptop with 256 Mbytes (2 Gbits) of memory. At this altitude, the soft error rate (SER) of 600 FITs per megabit becomes 100,000 FITs per megabit, resulting in a potential error every five hours.
  - 1 FIT (failures in time), is the number of failures in 1 billion device-operation hours. A measurement of 1000 FITs corresponds to a MTTF (mean time to failure) of approximately 114 years.

Vigade klassifikatsioon

- Püsivad: viga on stabiilne ja alaline
  - Vigane komponent tuleb asendada
  - Klassikaline näide: *stuck-at* vead
- Perioodilised rikked (intermittent)
  - Esinevad ajutiselt, seoses süsteemi või selle komponentide ebastabiilsusega (näiteks ebakindel ühendus)
- Ajutised rikked (transient)
  - Viga, mis lähtub ajutistest keskkonnamõjudest
  - Näiteks pingelang või EMI

Äikesetormid

Tõrgete klassifikatsioon

- ✓ Domain/Nature
  - Value failure
  - Timing failure
- ✓ Perception
  - Consistent failure
  - Inconsistent failure
- ✓ Effect
  - Benign failure
  - Malign/catastrophic failure
- ✓ Frequency
  - Single failure
  - Repeated failure

Tõrked (Failures)

- **Crash** Failure: After an error has been detected, the component stops silently.
- **Omission** Failure: Sometimes a result is missing; when result is available, it is correct.
- **Consistent** Failure: If there are multiple receivers, all see the same erroneous result.
- **Byzantine** (Malicious, Asymmetric) Failure: Different receivers see differing results.

Tõrked (2)

- **Timing** Failure: A server's response lies outside the specified time interval.
- **Response** Failure: The server's response is incorrect (value of the response is wrong, server deviates from the correct flow of control).
- **Arbitrary** Failure: A server may produce arbitrary responses at arbitrary times.

Vigadega toime tulemine

- Vigade ellimineerimine: eemalda probleemide algusallikad
  - Kõrvalda defektid: test & debug
  - Robustne disain: vähendab defektide tõenäosust
  - Vähenda keskkonnamõjusid: kaitsmed

Võimatu on vältida kõiki vigu

- Veakindlus: liiasuse lisamine, et vigu maskeerida
  - Vaja on täiendavaid ressursse
  - Näited:
    - Error correcting codes (Hamming, Red Solomon), hääletamine, maskeerimine, kontrollsummad...
    - Backup, replication, ...

Veakindluse saavutamine

- **Vigade avastamine** on protsess, et tuvastada vigade esinemist. Veakindluse saavutamise esmaseid tegevusi. Näiteks error detection codes, isekontrollivad loogikaskeemid, timerid, jne...
- **Vigade lokaliseerimine** on protsess, et tuvastada, kus viga on toimunud, et alustada vastava taastamisprotseduuriga

Veakindluse saavutamine

- **Vigade piiritlemine** on protsess, et viga isoleerida ja vältida selle mõju süsteemi ülejäänud osadele (vältida levimist)
- **Veast taastumine** on protsess, mille käigus süsteem püüab jääda tööle või taastada oma töövõime läbi rekonfiguratsiooni (isegi, kui rike on süsteemis alles). Näited: vigade maskeerimine, kordamine, tagasipöördumine jne...

Definitsioonid

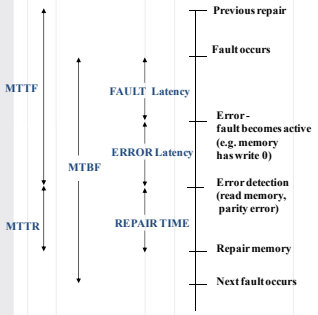
- Tõrgete sagedus (λ):
  - Keskmine sagedus, millega tõrked tekivad.

$$\frac{6\ failures}{7502\ hrs} = 0.0007998\ failures/hr = 799.8\times10^{-6}\ failures/hr$$

- Mean time to failure (MTTF):
  - Average time between failures

$$MTTF = \frac{1}{\lambda}$$

Usaldusväärsus



Reliability (Töökindlus):
a measure of the continuous delivery of service;
R(t) is the probability that the system survives (does not fail) throughout [0, t];
expected value: *MTTF (Mean Time To Failure)*

Maintainability (Remonditavus):
a measure of the service interruption
M(t) is the probability that the system will be repaired within a time less than t;
expected value: *MTTR (Mean Time To Repair)*

Availability (Töövalmidus):
a measure of the service delivery with respect to the alternation of the delivery and interruptions
A(t) is the probability that the system delivers a proper (conforming to specification) service at a given time t.
expected value: *EA = MTTF / (MTTF + MTTR)*

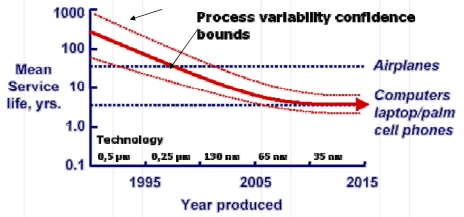
Safety (Ohutus):
a measure of the time to catastrophic failure
S(t) is the probability that no catastrophic failures occur during [0, t];
expected value:
MTTCF (Mean Time To Catastrophic Failure)

Tehnilised trendid

	1990	2000	2010
Operating temperature, °C	-55 to 125	-40 to +85	0 to 70
Supply voltage	5v	1.5v	0.6v
Max. power (high perf.)	5	100	170
No. of package types	<10	<60	??
Design support life	>10 yrs.	1-5 yrs.	<1yr.
Production life	>10 yrs.	3-5 yrs.	<3yrs.
Service life	***>20 yrs.***	***5-10 yrs.***	***<5yrs.***

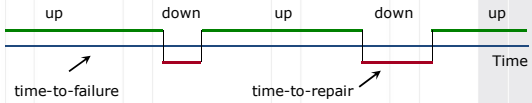
*MRQW-2002, Bernstein

Kiipide töökindluse hindamine



*Extrapolated from ITRS roadmap, MRQW-2002, Bernstein

Töövalmidus



$$Availabili\ ty = \frac{MTTF}{MTTF + MTTR}$$

- Töövalmidus:
  - Probability that system is operational at time *t*
- Kõrge töövalmidus:
  - MTTF → infinity (high reliability)
  - MTTR → zero (fast recovery)

Remonditavus

- *M(t)* on tõenäosus, et vigane süsteem on võimalik uuesti töökorda viia ajaperioodi *t* jooksul.
- Taastamise protsess:
  - Vea lokaliseerimine, näiteks läbi diagnostika
  - Süsteemi parandamine
  - Süsteemi töövalmidusse toomine

Graceful Degradation

- The ability of system to automatically decrease its level of performance to compensate for hardware failure and software errors.

Töövalmiduse üheksate müüt

Üheksaid	Töövalmidus	Rikkeaeg aastas	Rikkeaeg nädalas	Näide
2 üheksat	99%	3.65 päeva	1.7 tundi	Tavaline veebikülg
3 üheksat	99.9%	8.75 tundi	10.1 min	E-kaubandus
4 üheksat	99.99%	52.5 min	1.0 min	Suur mailiserver
5 üheksat	99.999%	5.25 min	6.0 s	Telefonisüsteem
6 üheksat	99.9999%	31.5 s	0.6 s	Carrier-grade andmeside

Ajalooline areng

- Mean Time Between Failures:

$$MTBF = MTTR + MTTF$$

  - ENIAC. MTBF: 7 minutit (18000 elektronlampi)
  - F-8 Crusader – esimene fly-by-wire
    - MD-11
    - A320 family
  - Patriot raketikaitse süsteem
    - Vajas rebooti iga 8 tunni järel. Sellest saadi teada väga valusal moel esimese lahesõja ajal...

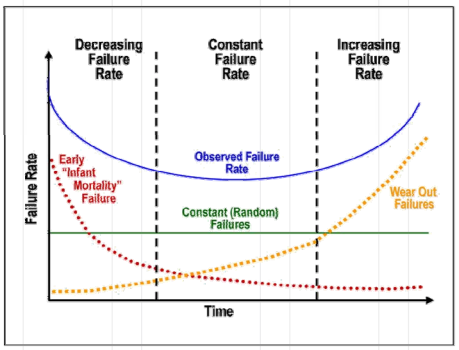
Ultra-töökindlad süsteemid

- Airbus A320 perekonna fly-by-wire süsteem:
  - Kõik täiturid on arvutite poolt kontrollitavad
  - Ei mingeid juhttrosse ega kaableid
  - 5 keskset lennuarvutit
  - Kasutatakse erinevaid riistvaralisi lahendusi
    - Thomson CSF => 68010
    - SFENA => 80186
  - Tarkvara mõlemale platvormile on kirjutatud erinevate tarkvaraloojate poolt (sõltumatult)
  - Kogu rikete avastamine & debug teostati eraldi
  - Arvuti lubab piloodil lennata mingite ettenähtud parameetrite piires (flight envelope)
    - väljaspool seda võtab arvuti juhtimise üle

Riistvara ja keskkonna tõrked

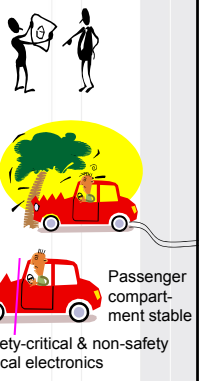
- Liikuvad osad, suur kiirus, madal tolerants, suur keerukus: kettad, tape drives/libraries
- Madalaim MTBF on ventilaatoritel ja toiteallikatel
- Tihti ventilaatorid "väsivad" → väikesed juhuslikud rikked CPUs, mälus, backplane'is
- Keskkond: Vool, jahutus, niiskus, kaablid, tuli, kokku kukkuvad rackid, ventilatsioon, tormid, ...

Vanni kõver



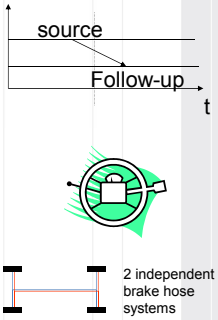
Kopetz'i 12 disainipõhimõtet (1-3)

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.



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.



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.

Kopetz'i 12 disainipõhimõtet (10)

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

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.

Valideerimine

Valideerimine

- Valideerimine on kontroll, et loodud süsteem vastab talle pandud piirangutele, töötab nagu eeldatud – kas me lõime õige asja.
- Kasutatakse palju simuleerimist, emuleerimist, ...

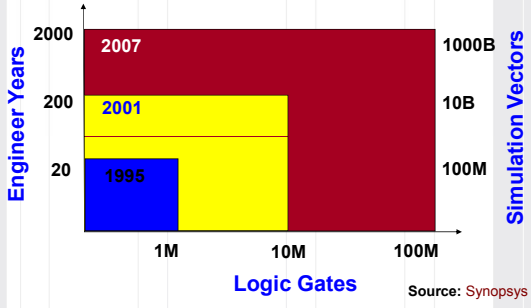
Simuleerimine

- Simuleerimine üritab imiteerida tegeliku süsteemi käitumist arvutis
- Funktsionaalse käitumise simuleerimine eeldab täidetava mudeli olemasolu
- Simuleerimist on võimalik teostada erinevatel abstraktsioonitasemetel
- Simuleerida on võimalik ka mittefunktsionaalseid omadusi (näiteks temperatuur, EMC jne.)
- Simuleerimist on võimalik kasutada disaini evalveerimiseks ja valideerimiseks

Funktsionaalne valideerimine kasutades simuleerimist

- Kasutatakse erinevaid abstraktsioonitasemeid
  - Kõrge tase: kiire aga ebatäpne
  - Madal tase: aeglane aga üldjuhul täpne
  - Õige taseme valimine on alati kompromiss

Kiipsüsteemide funktsionaalne valideerimine



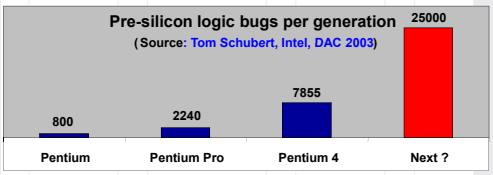
Source: Synopsys

71% of SOC re-spins are due to logic bugs

Source: G. Spirakis, keynote address at DATE 2004

Mikroprotsessorte funktsionaalne valideerimine

- Functional validation is a major bottleneck
  - Deeply pipelined complex microarchitectures



- Logic bugs increase at 3-4 times/generation
  - Bugs increase (exponential) is linear with design complexity growth.

Termoanalüüs ja termomudelid

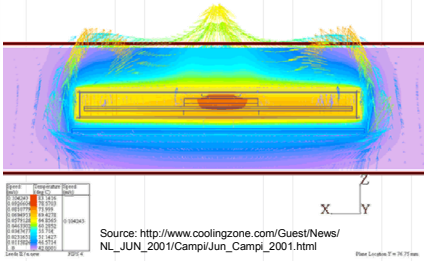
- Termomodelleerimine on muutumas üha olulisemaks
  - Kuna temperatuurid on suureneva jõudluse tingimustes oluliselt kasvanud
  - Kuna temperatuur mõjutab
    - Kasutatavust
    - Töökindlust

Mudeli osad

- Termojuhtivus kirjeldab läbi materjali pindlaga *A* ja pakusega *L* leviva kuumuse hulka, kui materjali erinevate poolte temperatuuride vahe on 1 Kelvin
- Termojuhtivuse vaste on termotakistus
- Termotakistus on summeeruv nagu elektriline takistus
- Kuumust talletavat massi võib võrrelda kondensaatoriga
- Termomodelleerimine põhineb üldjuhul analoogilistel elektrimudelitel ja kasutab tuntud elektrivõrgu lahendusalgoritme

Näide termosimuleerimisest

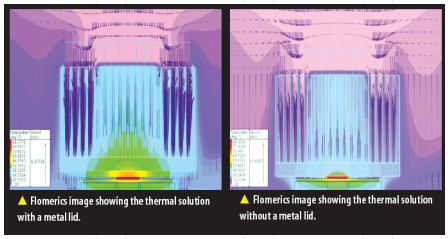
- Encapsulated cryptographic coprocessor:



Source: http://www.coolingzone.com/Guest/News/NL_JUN_2001/Campi/Jun_Campi_2001.html

Näide termosimuleerimisest (2)

Mikroprotsessor



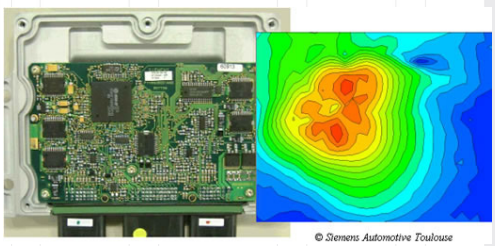
▲ Flomerics image showing the thermal solution with a metal lid.

▲ Flomerics image showing the thermal solution without a metal lid.

Source: http://www.flotherm.com/applications/app141/hot_chip.pdf

EMC simuleerimine

Näide: auto mootorikontroller (ECU)



Red: high emission
Validation of EMC properties often done at the end of the design phase.

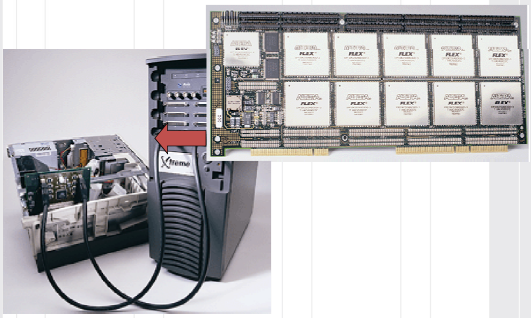
Source: http://intrage.insa-tlse.fr/~etienne/emccourse/what_for.html

Simuleerimise piirangud

- Tüüpiliselt aeglasem kui valmis disain.
  ☞ Ajaliste piirangute mittetäitmine on seetõttu reaalses keskkonnas väga tõenäoline
- Simuleerimine reaalses keskkonnas võib olla ohtlik
- Simuleeritavate andmete hulk võib olla tohutu
- Enamus reaalseid süsteeme on liiga keerukad, et simuleerida kõike (sisendeid)
  Simulatsioonid aitavad vigu leida, kuid need ei garanteeri vigade mitteesinemist



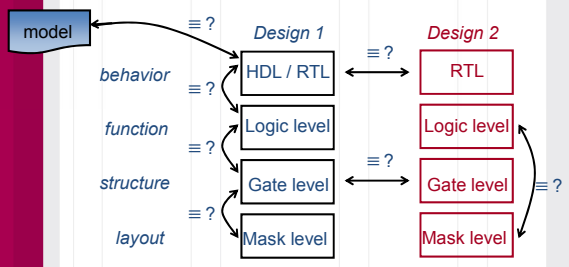

Emuleerimine



- [www.verisity.com/images/products/xtremep{1|3}.gif]

Verifitseerimine

Verifitseerimine



Verifitseerimine

- Põhiline eesmärk on kontrollida, kas meie tehtud disainisammu tulemus on õige. Kas me disainisime süsteemi õieti.
  - Spec→süsteemitase, kõrgtase→RTL, jne...
  - Simuleerimine, emuleerimine, ...
- Formaalne verifitseerimine
  - Formaalne kontroll, kasutades formaalseid mudeleid, matemaatikat (loogikat)
    - Model checking, equivalence checking

Ekstreemne näide

- How do you verify a design which has bugs like this??

- The FMUL instruction, when the rounding mode is set to "round up", incorrectly sets the sticky bit when the source operands are:
  src1[67:0] = X*2i+15 + 1*2i
  src2[67:0] = Y*2j+15 + 1*2j
  where i+j = 54 and {X,Y} are integers

And the answer is...

- Hire 70+ validation engineers
- Buy several thousand compute servers
- Write 12,000 validation tests
- Run up to 1 billion simulation cycles per day for 200 days
- Check 2,750,000 manually-defined properties
- Find, diagnose, track, and resolve 7,855 bugs
- Apply formal verification with 10,000 proofs to the instruction decoder and FP units
  - This found that obscure FMUL bug!

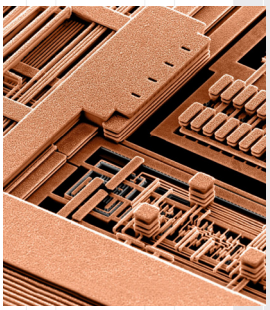
Testimine

Testimine

- Testimine tootmisvigade vastu:
  - Tootmisliini lõpus
  - Peale tarnimist
- O&M test
  - Peale tarnimist
- Test: testide genereerimine, testide rakendamine, väljundite jälgimine, väljundi hindamine

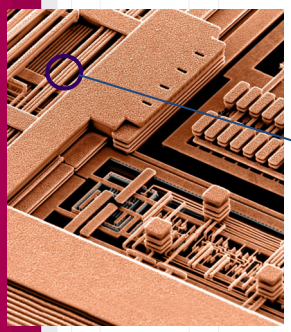
Defektid

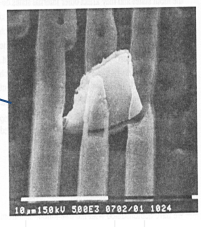
- Device level
  - shorts, cracks, leaks, impurities, bonding, ...
- Board level
  - component errors, track errors, ...
- Functional
  - Incorrect design
- Wearout/Environment
  - temperature, humidity, vibration, radiation...



Fault model -
abstraction mechanism for describing defects mathematically

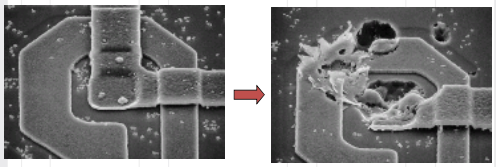
Näide tootmisdefektist





Näide vananemisest

Metal migration @ Pentium 4



www.jrwhipple.com/computer_hangs.html

Veamudelid

- Stuck-at fault model

Possible Errors: 6
"a" s-a-1, "a" s-a-0
"b" s-a-1, "b" s-a-0
"c" s-a-1, "c" s-a-0

Testide genereerimine

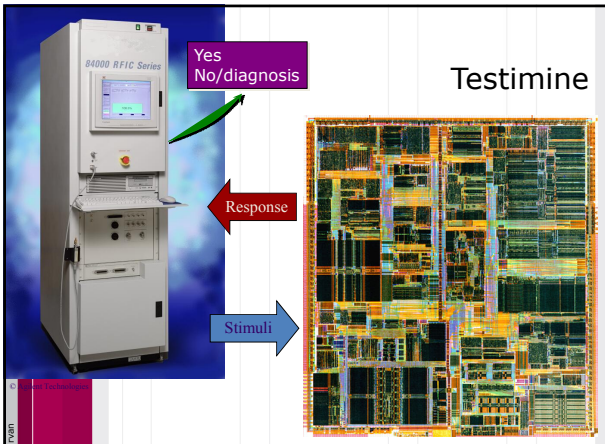
- Totaalne testimine

  2^n test vektorit n sisendiga kombinatoorse skeemi jaoks

n	2	10	20	100
2^n	4	$\sim 10^3$	$\sim 10^6$	$\sim 10^{30}$

- Pseudojuhuslikud testid
- Deterministlikud testid
  - **Deterministlike testide genereerimine on NP-keerukas probleem!!**
  - Erinevad heuristikad

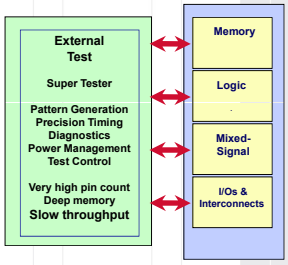
Testimine



Support from
Automatic Test Equipment

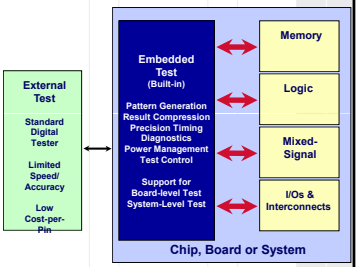
Väline Testimine

- Probleemid
  - ATE-d on väga kallid (tüüpiliselt mitu miljonit USDd)
  - ATE-d muutuvad üha ebaeffektiivsemateks
  - Aeglane testide rakendamine
  - Andmemahud võivad olla väga suured (sõltuvuses kiibi keerukusest)



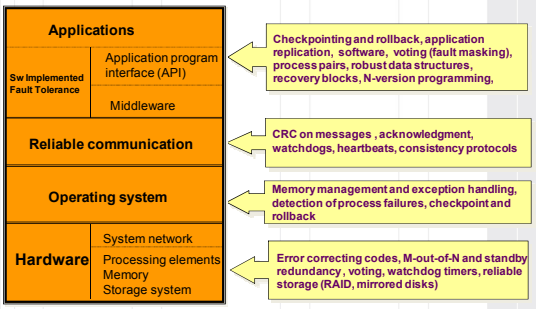
Built-in Self-Test (BIST)

- Lahendus: spetsiaalne sisse-ehitatud riistvara testide genereerimiseks ja rakendamiseks
- Testri tükeldamine
- Odav aga suure kiirusega sisemine tester



Kokkuvõte

Töökindlus nõuab süstemaatilist lähenemist



[Iyer]

Veakindlus

- Veakindlus ja usaldusväärsus on süsteemsed teemad, nõudes töötamist nii riistvara, tarkvara, aja kui ka informatsiooni probleemidega
- Üha keerukam on töötada riistvara probleemidega

Küsimusi?

Gert Jervan
ati.ttu.ee/~gerje