1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

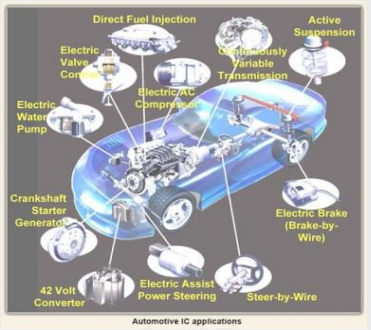
Department of Computer Engineering
ati.ttu.ee

Süsteemide diagnostika
IAF0250

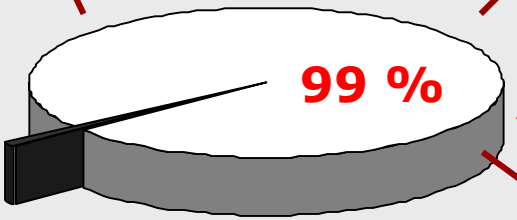
Raimund Ubar
raiub@pld.ttu.ee
Arvutisüsteemide instituut

Tänane IT-põhine tehismaailm

Universaal-arvutid 1%



Embedded systems 99%



Mikroprotsessorite osakaal

Me märkame oma sõltuvust infotehnoloogiast alles siis kui midagi ootamatult ei tööta

Mida sellest kursusest õppida võib

Diagnostika kui õppeaine

- tähendab intellektuaalset harjutust põhjuste ja tagajärgede mõtestamisel
- õpetab formuleerima õigeid küsimusi
- õpetab planeerima eksperimente ning analüüsima eksperimentide tulemusi

Mida sellest kursusest õppida võib

Mille poolest erinevad

- ✓ **tarkvarainsener ja**
- ✓ **arvutiinsener,**
  kes on õppinud süsteemide diagnostikat?

Tarkvarainsener oskab programmeerida arvutit, mis töötab

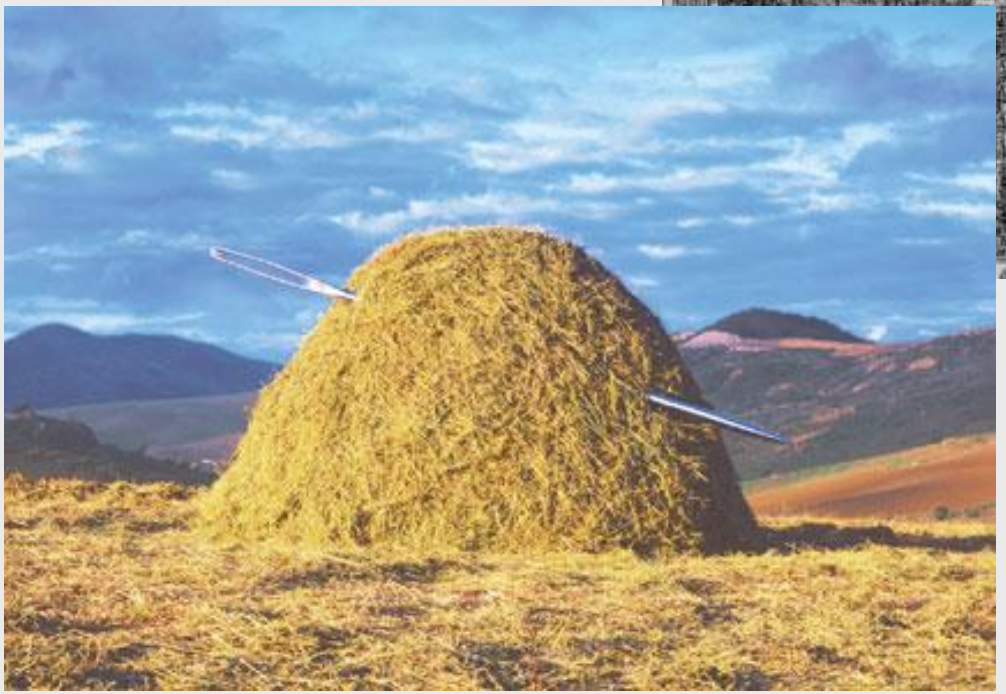
Diagnostikat õppinud insener oskab programmeerida arvutit, mis ei tööta

Mida sellest kursusest õppida võib

Tänaste süsteemide iseloomulikuks tunnuseks on keerukus

Kursus juhatab sisse strateegiatesse, mis aitavad jagu saada süsteemide keerukusest:

- abstraheerimine
- modelleerimine
- simuleerimine
- optimeerimine
- paralleliseerimine



Kursus õpetab,
kuidas
leida nõela
heinakuhjast

Süsteemide projekteerimise kaks poolt

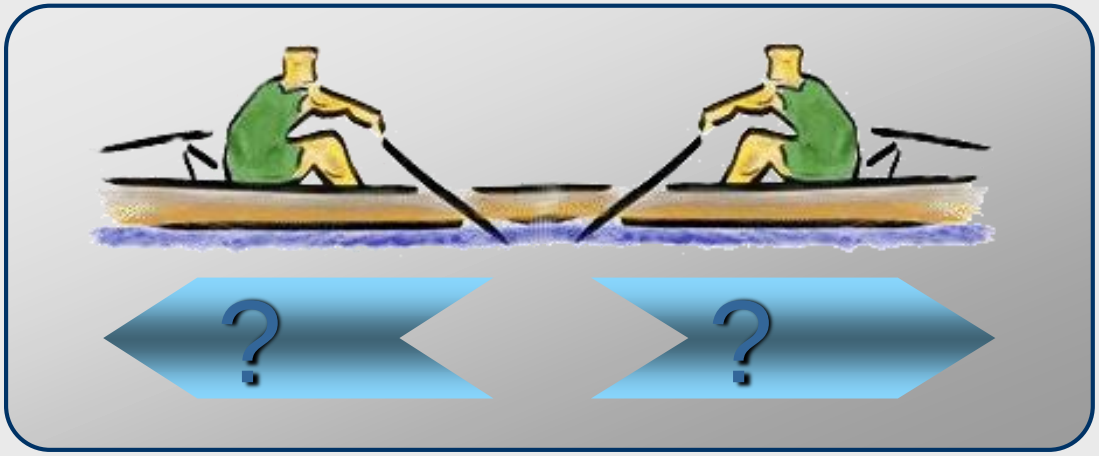
Disainer diagnostikainsenerile:

- ✓ Kontrolli, kas mu süsteem on töökorras

Diagnostikainsener vastab:

- ✓ Disaini uuesti, nii et kõikidele vajalikele signaalidele oleks testimisel ligipääs

Dialoog:



H.-J. Wunderlich, U Stuttgart

Paradigma muutus: ***Design for testability***

Põhieesmärk - süsteemide usaldusväärsus

Süsteemid meie ümber, moodustavad **tehismaailma**, mille **usaldusväärsusest** me üha enam sõltume

Joonistus: Tiina Ubar

„Süsteemide diagnostika"

Süsteemide diagnostika

1. Sissejuhatus

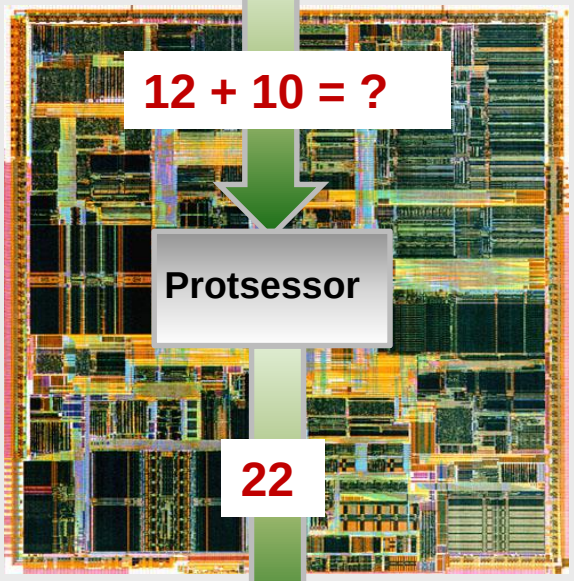
- 1.1. Kursuse eesmärgid ja probleemi püstitus
- 1.2. Kursuse sisu ja struktuur
- 1.3. Süsteemide diagnostika põhiprobleemid
- 1.4. Kursusetöö ja bakalaureusetöö temaatikast

Kursuse eesmärgid

- Õppida tundma digitaalskeemide ja süsteemide filosoofiat, süsteemide rikete põhjusi, nende avastamise ja diagnoosimise meetodeid
- Saada ülevaade ja teoreetiline kogemus
  - testimise põhimõistetest ja rikete modelleerimisest
  - testide sünteesi ja analüüsi meetoditest
- Õppida
  - praktiliselt kasutama professionaalset eritarkvara
  - analüüsima ja võrdlema eri meetodite efektiivsust, arendama oma kriitilist meelt
  - orienteeruma testimise majanduslikes aspektides nagu testimise kvaliteet ja hind, kompromisside tegemine kvaliteedi ja hinna vahel

Mis on test?

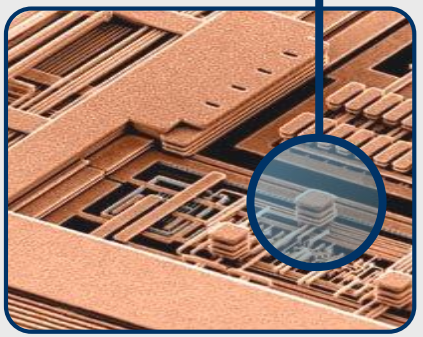
Test-programm



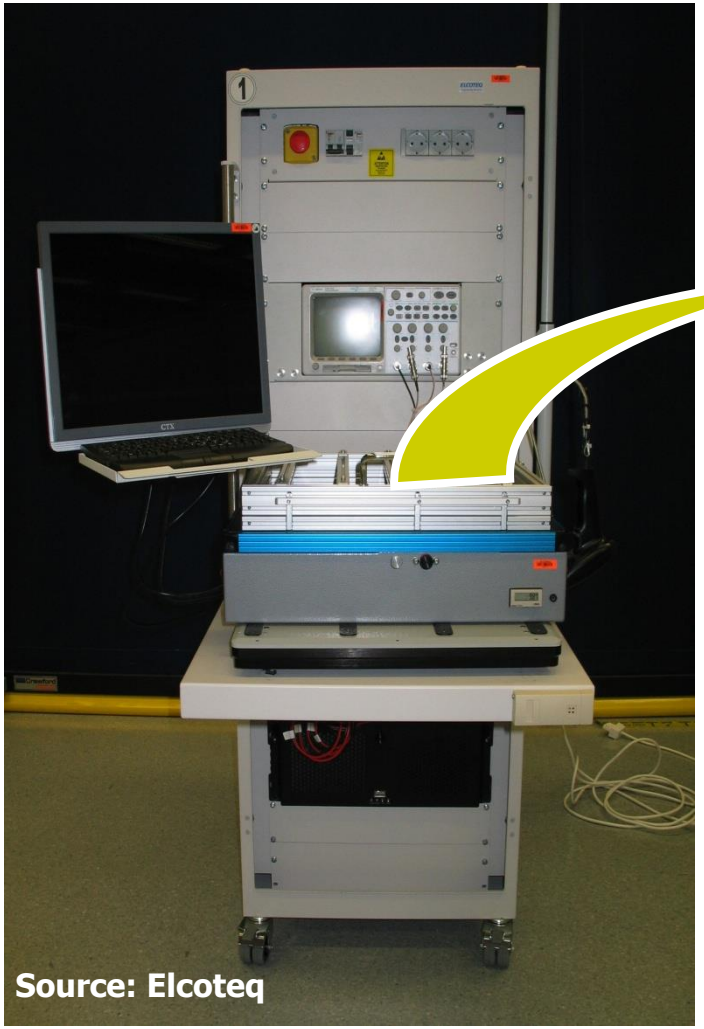
Testi tulemused

Mitme operandiga peaks testima summaatorit, et veenduda selle korrasolekus?

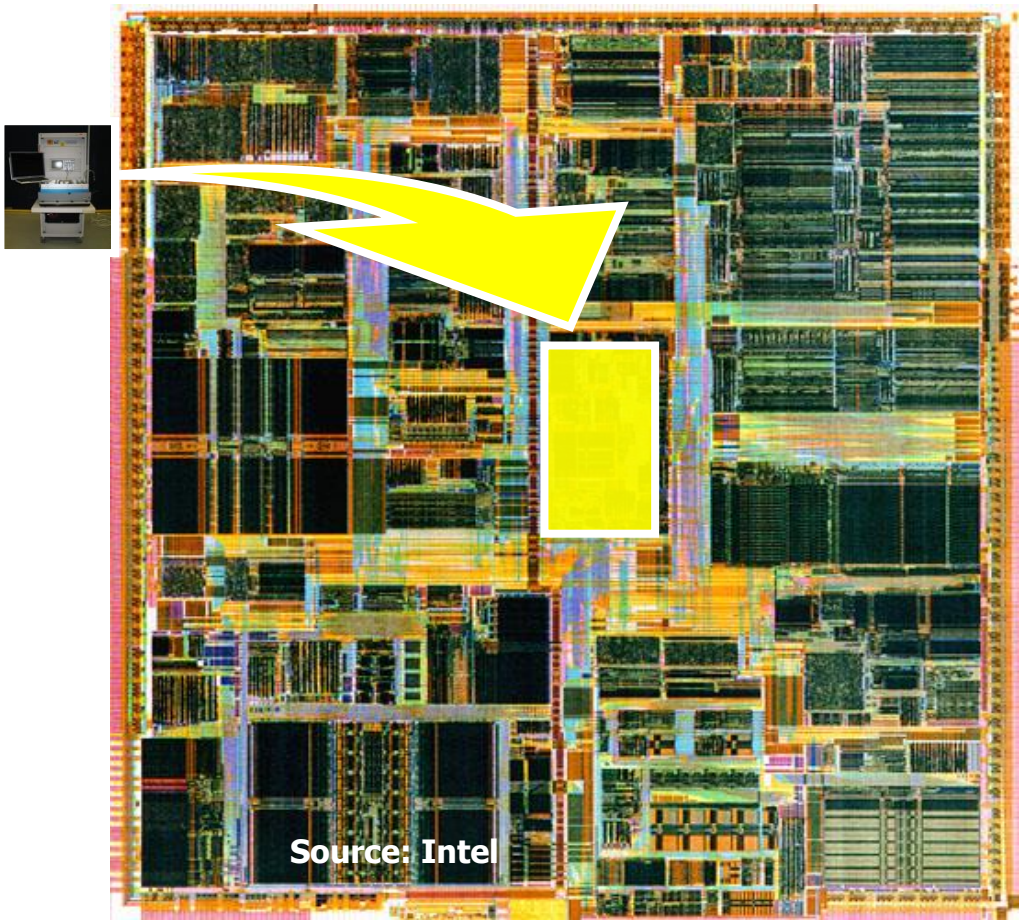
Diagnoos



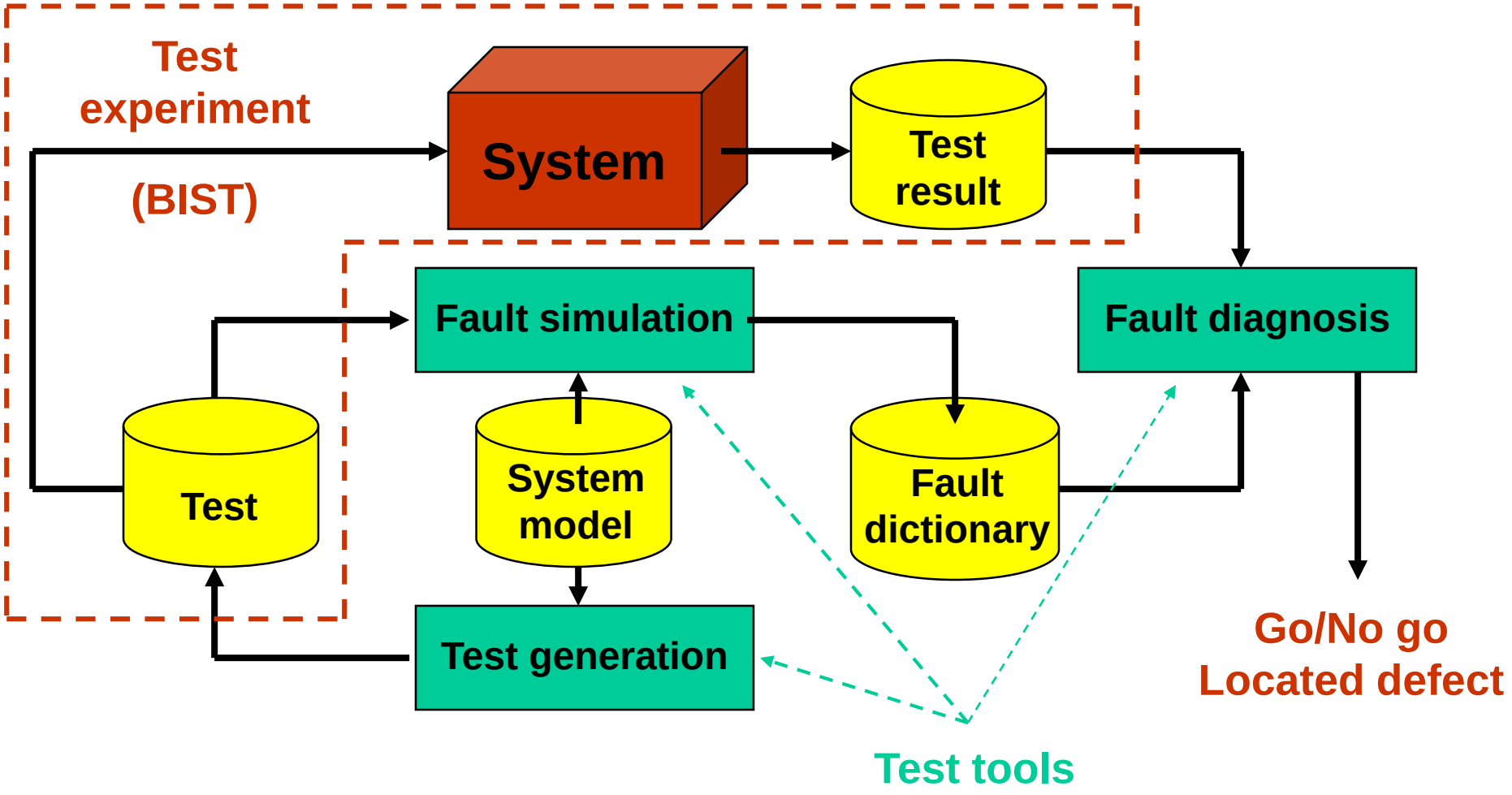
Built-in Self-Test (BIST)



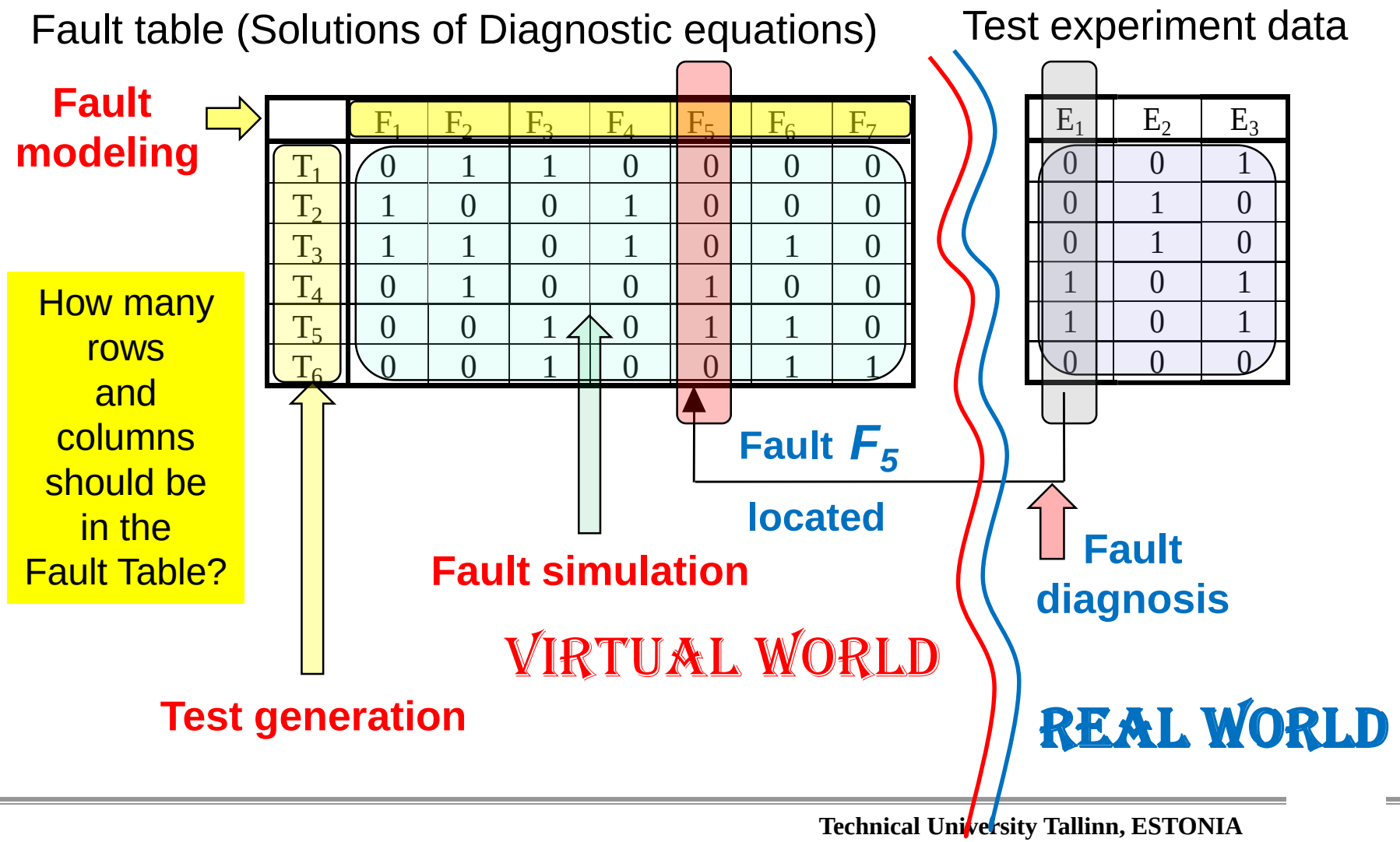
Cores have to be tested on chip



The Test World: Tools, Data and Testing



Test Related Basic Problems



Diagnostika kui disaini pöördprobleem

X

$F(X)$

Y, dY

Süsteemi mudel:

$$Y = F(X)$$

Diagnostika üldvõrrand:

$$dY = F(X,dX)$$

Diagnoos:
(pöördülesanne)

$$dX = F^{-1}(X,dY)$$

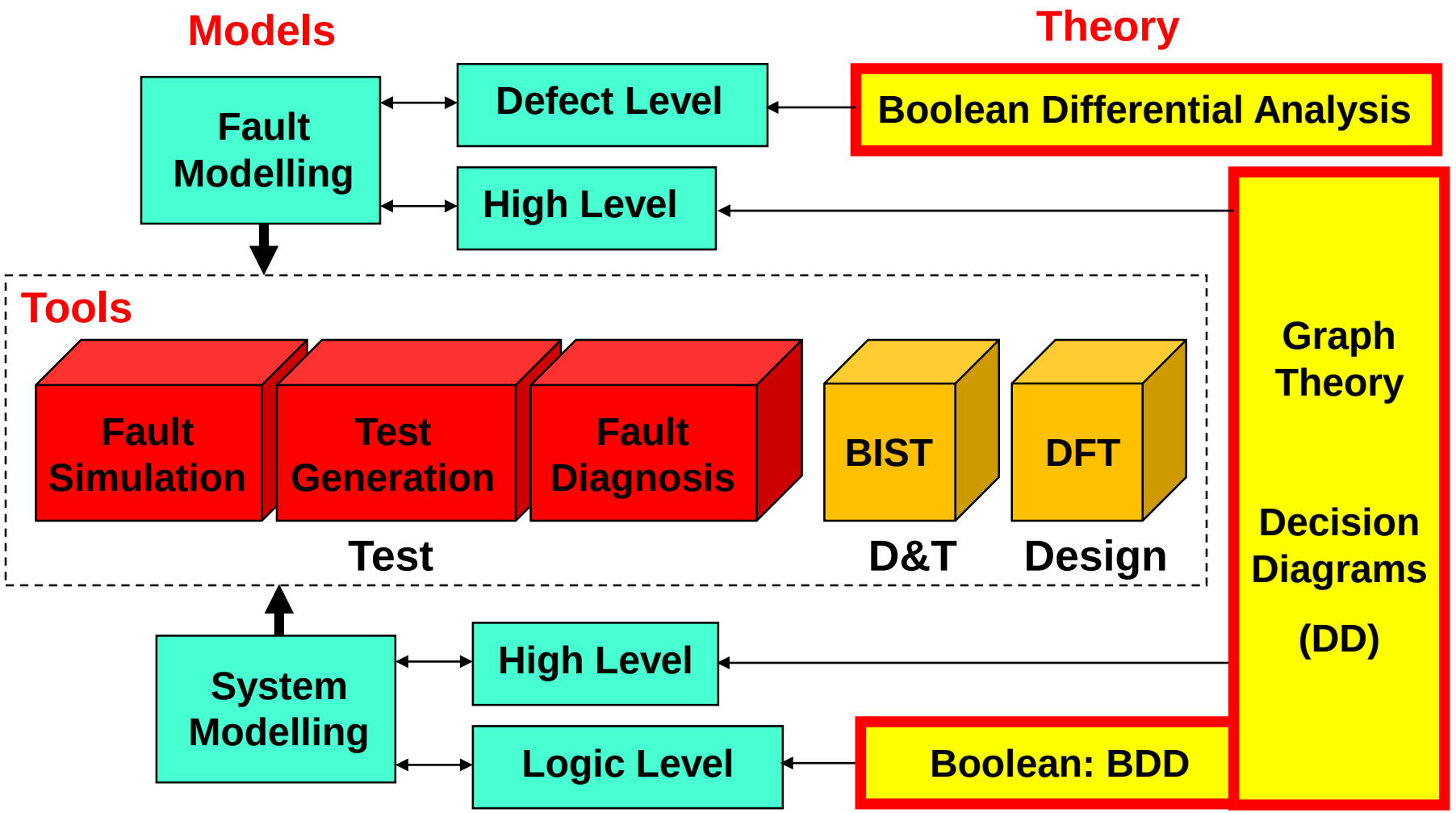
Ülesanne	Antud	Leida
Testide süntees	dX	X
Testide analüüs	X	dX

Süsteemide diagnostika

1. Sissejuhatus

- 1.1. Kursuse eesmärgid ja probleemi püstitus
- 1.2. Kursuse sisu ja struktuur
- 1.3. Süsteemide diagnostika põhiprobleemid
- 1.4. Kursusetöö ja bakalaureusetöö temaatikast

Introduction to Theories: The Course Map



Kursuse sisu

1. Sissejuhatus
- 1.1. Kursuse suur pilt
- 1.2. Kiire pilguheit põhiprobleemidesse

2. Teoreetilised alused
- 2.1. Boole'i differentsiaalalgebra
- 2.2. Binaarsed otsustusdiagrammid (BDD)
- 2.3. Kõrgtasandi otsustusdiagrammid

3. Rikete modelleerimine
- 3.1. Rikete klassifikatsioon
- 3.2. Loogikatasandi konstantrikked
- 3.3. Tingimuslikud rikked
- 3.4. Kõrgtasandi rikked

4. Testide süntees
- 4.1. Deterministlik testide süntees kombinatsioonskeemidele
- 4.2. Testide genereerimine otsustusdiagrammide abil
- 4.3. Triviaalsete (pseudotäielike) testide süntees
- 4.4. Testide süntees kordsetele riketele (üldjuht)
- 4.5. Testide süntees digitaalsüsteemidele kõrgtasandil

5. Testide analüüs
- 5.1. Rikete simuleerimise meetodid
- 5.2. Testide analüüs deduktiivsete meetoditega
- 5.3. Rikete simuleerimine mäluga skeemides
- 5.4. Testide analüüs kõrgtasandil

6. Rikete diagnoos
- 6.1. Rikete tabelid ja puud
- 6.2. Kombinatoorne diagnoos (diagnostikasõnastikud)
- 6.3. Sekventsiaalne ehk adaptiivne diagnoos

Kursuse üldstruktuur

- ✓ Loengud (1x nädalas)
- ✓ Laborid/harjutused (1x nädalas) – alates 3. nädalast
- ✓ 3 laboritööd
- ✓ Laborite arvestus
- ✓ Iseseisev kursusetöö (esitatakse ja kaitstakse eksamil)
- ✓ Eksam
  - Iseseisva töö arvestus
  - 2 ülesannet
  - 1 teoreetiline küsimus

Õppekirjandus

1. M.L.Bushnell, V.D.Agrawal. Essentials of Electronic testing. *Kluwer Acad. Publishers*, 2013.
2. L.-T.Wang, C.-W.Wu, X.Wen. VLSI Test Principles and Architectures. Elsevier, 2006, 777 p.
3. **R.Ubar**. Digitaalsüsteemide diagnostika I. Diagnostiline modelleerimine. TTÜ Kirjastus, 148 lk., 2005 (II osa Internetis)
4. **R.Ubar**, A.Jasnetski, A.Tsertov, S.Oyeniran. Software-Based Self-Test With Decision Diagrams for Microprocessors, 174 p., 2018 (Manuscript in Internet)
5. **R.Ubar**, J.Raik, T.Vierhaus „Design and Test Technology for Dependable Systems-on-Chip ". IGI Global, USA, 2011
6. O.Novak, E.Gramatova, **R.Ubar**. Handbook of Testing Electronic Systems. Czech TU Publishing House, 2005, 395 p.

Õppekirjandus (2)

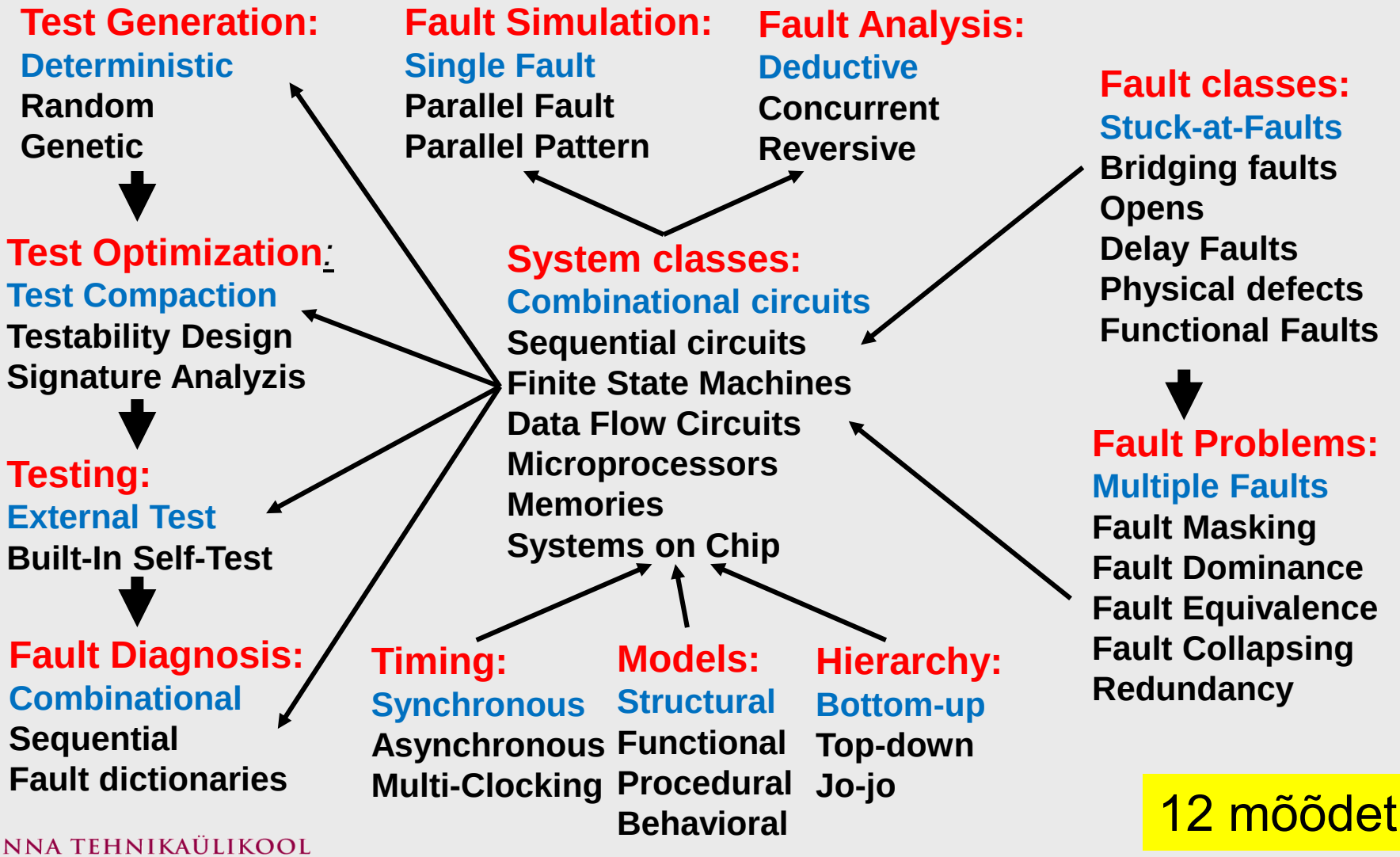
1. A.Miczo. Digital Logic Testing and Simulation. Wiley-Interscience, New Yersey, 2003, 668 p.
2. N.Jha, S.Gupta. Testing of Digital Systems. Cambridge Univ. Press, 2003, 1000 p.
3. H.-J.Wunderlich, Ed. Models in Hardware Testing. Springer, 2010
4. R.Drechsler, S.Eggersglüss, G.Fey, D.Tille. Test Pattern Generation using Boolean Proof Engines. Springer, 2009, 192 p.
5. D.K. Pradhan, I.G.Harris, Eds. Practical Design Verification. Cambridge Univ. Press, 2009, 276 p.
6. D.Gizopulos. Advances in Electronic Testing, Technology & Engineering. Springer, 2006.
7. D.Gizopulos, A.Paschalis, Y.Zorian. Embedded Processor-Based Self-Test. *Kluwer Acad. Publishers*, 2004, 216 p.

Süsteemide diagnostika

1. Sissejuhatus

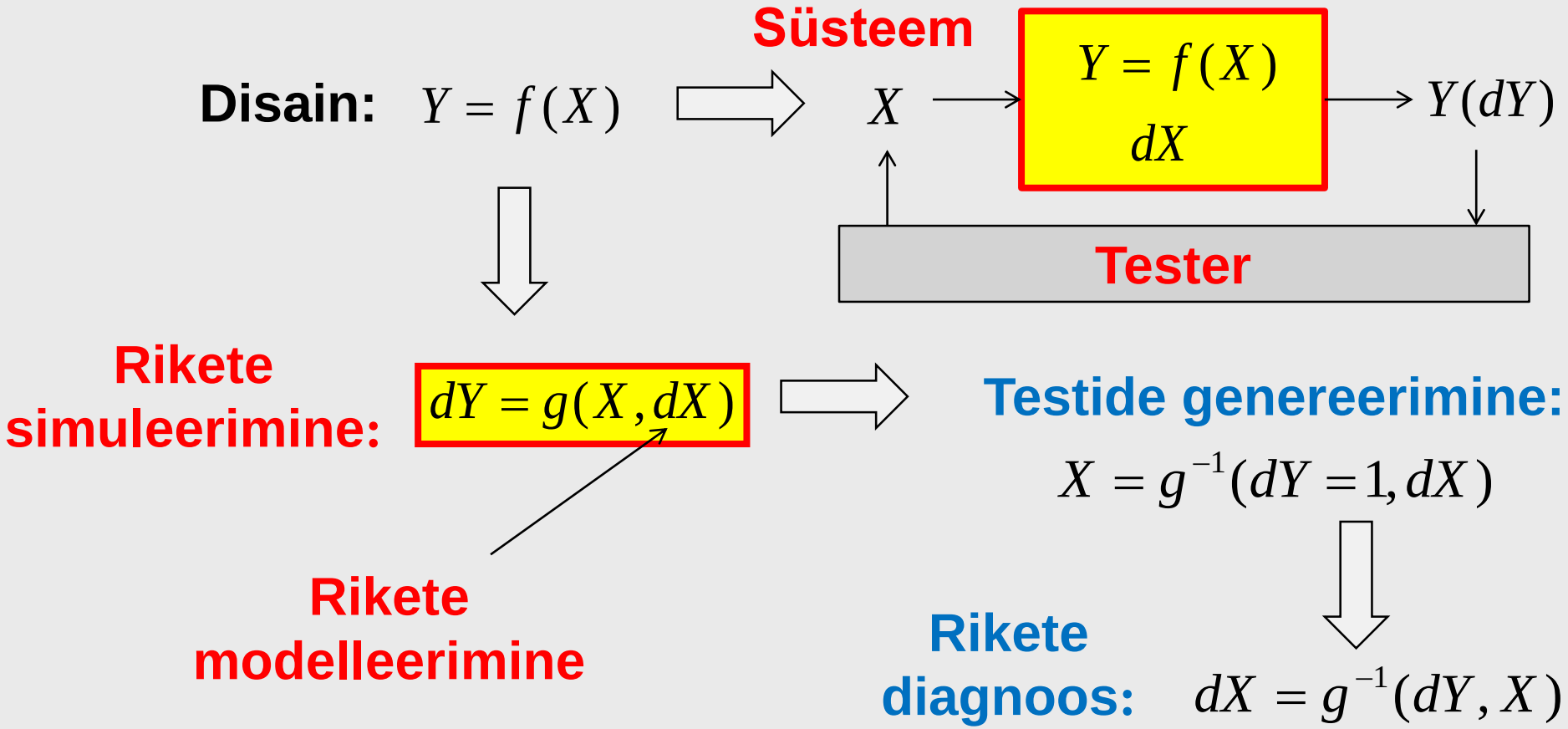
- 1.1. Kursuse eesmärgid ja probleemi püstitus
- 1.2. Kursuse sisu ja struktuur
- 1.3. Süsteemide diagnostika põhiprobleemid
- 1.4. Kursusetöö ja bakalaureusetöö temaatikast

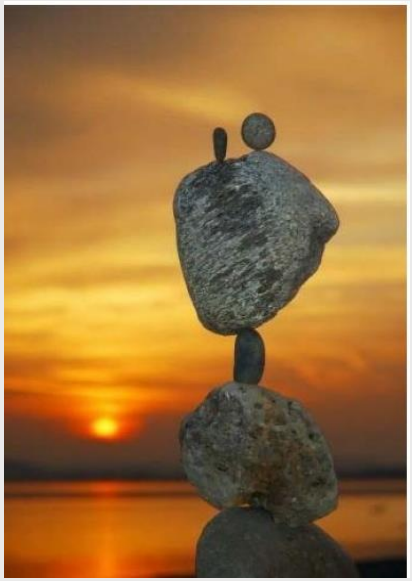
Diagnostika probleemide n-mõõtmeline ruum



12 mõõdet

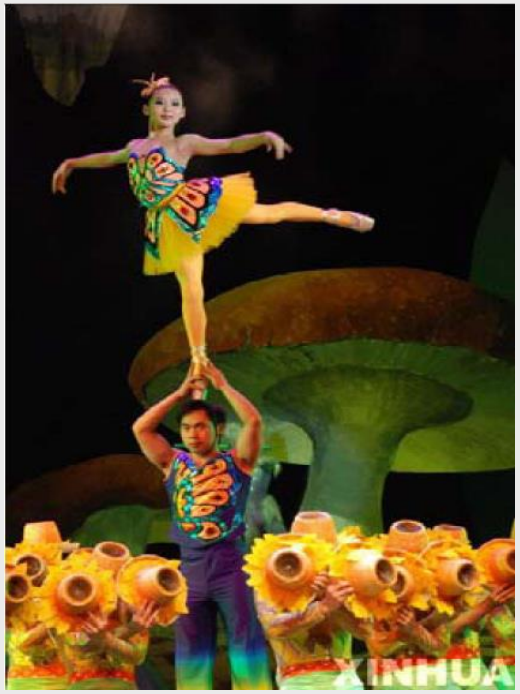
Kõik probleemid taanduvad ainsale võrrandile

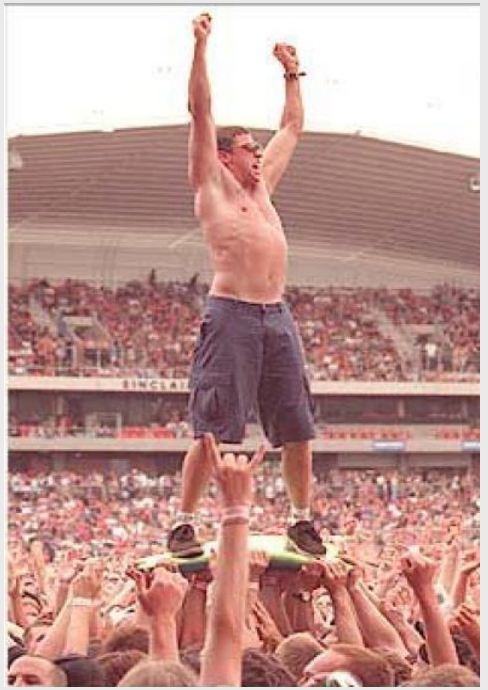




Põhieesmärk: veakindlus

- ✓ Nanotehnoloogia ajastu algab 100 nm transistori mõõtmest, täna on see alla 10 nm
- ✓ **Ebatöökindlatest elementidest on vaja teha töökindlaid süsteeme**

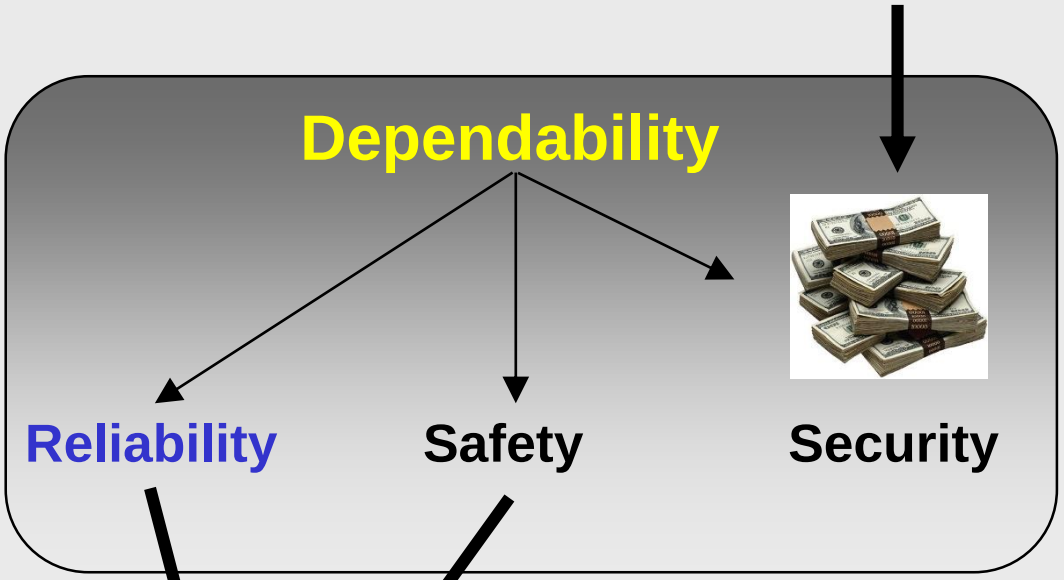




© Z.Peng, U Linköping

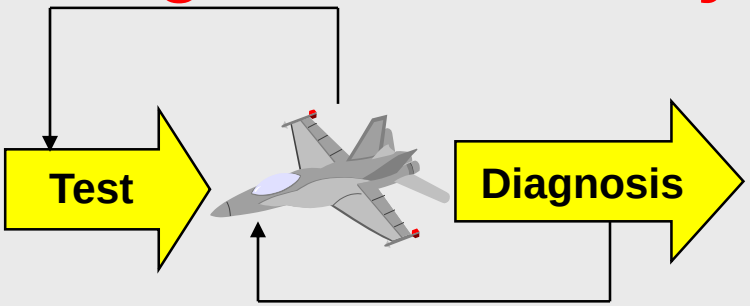
- ✓ Ei saa vältida maavärinate puudumist, ja on paratamatu seegi, et materjal väsib ja vananeb ning tekivad tõrked tehnilistes süsteemides
- ✓ Aga nii nagu loodus on andnud inimesele immuunsussüsteemi, nii on võimalik luua vahendeid immuunsuse (robustsuse) loomiseks ka tehissüsteemides

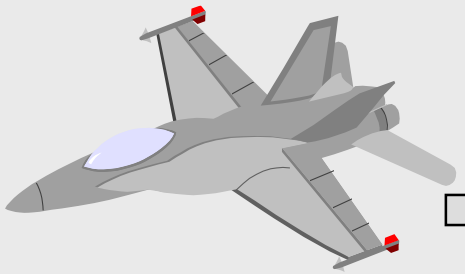
Dependability of Systems



There is no *security*
on the earth,
there is only oportunity
Douglas McArthur
(General)

Design for testability:

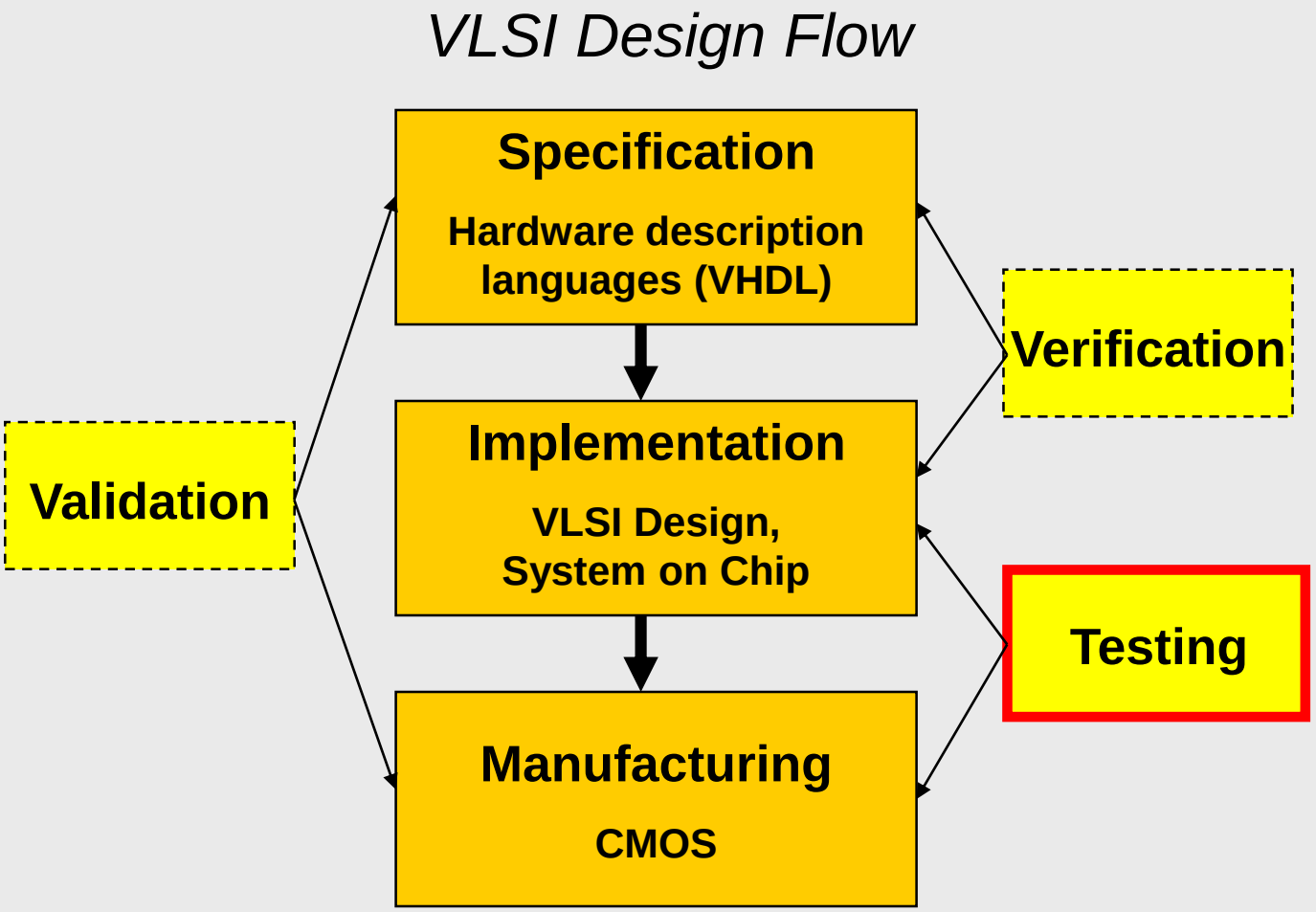




Test
Fault Diagnosis
BIST/BISD – Built-In Self-Test/Diagnosis
Fault-Tolerance
Self-Repair

Dependability

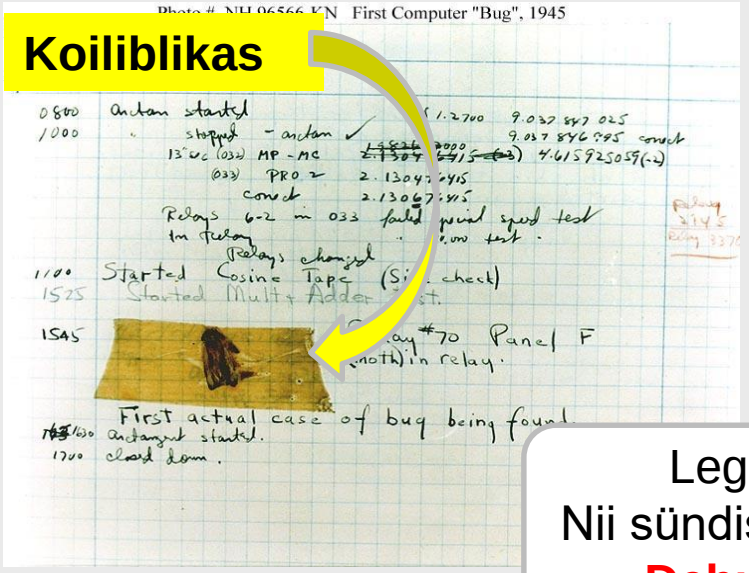
Terminology: Verification, Validation, Testing



Verification is to check the consistence between the individual development phases

Validation is checking the system whether it conforms to the user requirements

Diagnostika ajaloost



- ✓ **1945.** Esimene arvutiviga maailmas:
  - Koi sattub Harwardis arvuti Mark II kahe relee vahele
  - Logiraamatusse kirjutati: ***"Hey, we actually found a bug that was a real bug!"***

Legend:
Nii sündiski termin
Debuging

- ✓ **1995.** Intel Pentiumi protsessor jääb vahele jagamisega
  - Jagamise algoritm kasutas tabelit, milles oli 1066 arvu. Tabeli laadimisprogrammis oli tsükliviga, mille tõttu kirjutati tabelisse 5 arvu vähem

Viie aastaga on kasvanud USA-s kahjum arvutivigadest **5 korda**, ulatudes praegu **60 miljardi dollarini** aastas

Väljakutsed selles valdkonnas

- ✓ Süsteemide usaldatavuse, töökindluse ja turvalisuse määrab eelkõige elektroonika:
- ✓ Vanasõna „Usalda, aga kontrolli“ kehtib mitte ainult inimeste, vaid ka kogu tehismaailma kohta
- ✓ **TESTIMINE**
- ✓ Tähendab rikete ja vigade avastamist, ning üles leidmist,
- ✓ nende põhjuste otsimine on
- ✓ **DIAGNOSTIKA**

„Süsteemide diagnostika“

Kogu innovatsioonist auto juures kuulub 90% elektroonikale

Eurodest ja dollaritest ning autost

- ✓ Planeedil on hinnatud olevat kümneid miljardeid sardsüsteeme.
  - **Iga sardsüsteem on probleem** (ühes autos võib olla 100 mikroprotsessorit)
- ✓ **40%** auto hinnast hõlmavad **elektroonika** ja tarkvara
  - 20-30% auto hinnast kulub tarkvarale
  - 15-20% kulub riistvarale
- ✓ Kui palju maksab tarkvara loomine:
  - Üks rida koodi maksab ca **15-40 USD**
    - aga kaitsetööstuses – 100 USD
    - kosmoselaeva puhul juba – **1000 USD**

Testimisele ja vigade otsimisele kulub 20-25% auto hinnast

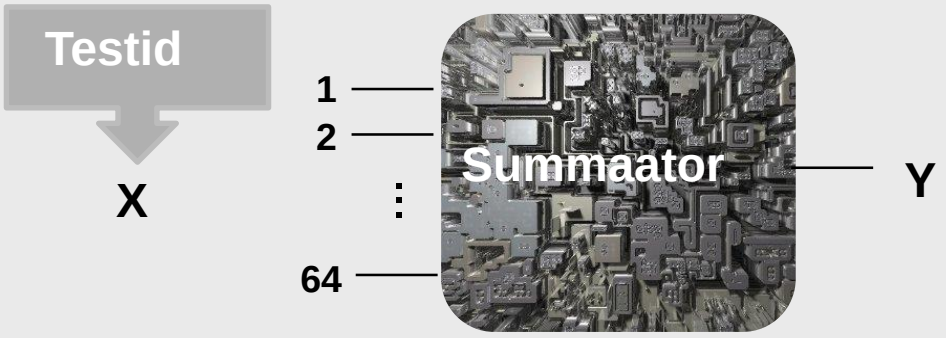
- ✓ **Tarkvara** projekteerimise kuludest läheb **50%** vigade otsimisele ja kõrvaldamisele
- ✓ **Riistvara** projekteerimise ja valmistamise kuludest läheb **70%** testimisele ja diagnostikale

Väljakutsed selles valdkonnas - Testimine

„Süsteemide diagnostika"

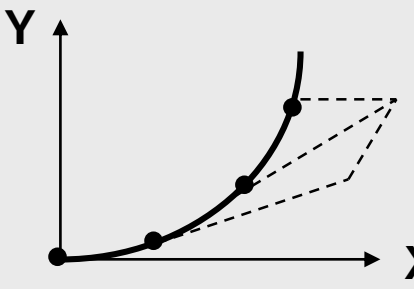
- ✓ Diagnostika – on **loogikaprobleem**
- ✓ Kõige keerulisem loogika on digitaalsüsteemide loogika
- ✓ Seepärast on diagnostikat õppida kõige loomingulisem **digitaalsüsteemide** näitel
- ✓ Abstraktsioon ja reaalsus on digitaalsüsteemides peaaegu üksüheses vastavuses

Digitaalsüsteem:



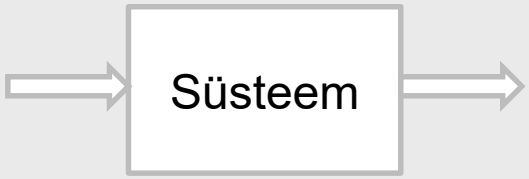
$X \rightarrow 2^{64} = 18\,446\,700\,000\,000\,000\,000 \approx 10^{19}$

Analoogsüsteem (võimendi):

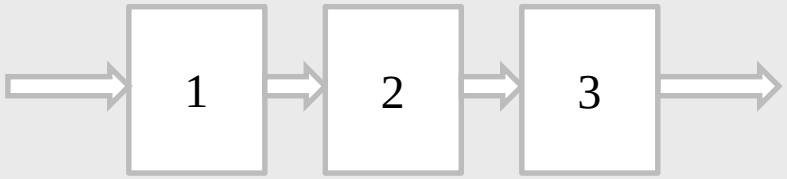


Kolmest mõõtmisest piisab

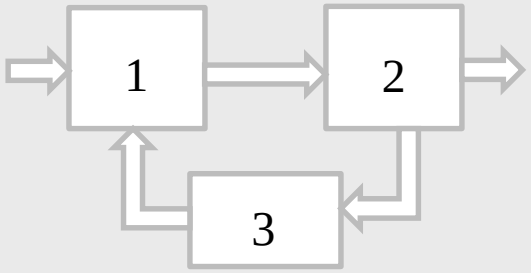
Millest tuleneb diagnostika keerukus?



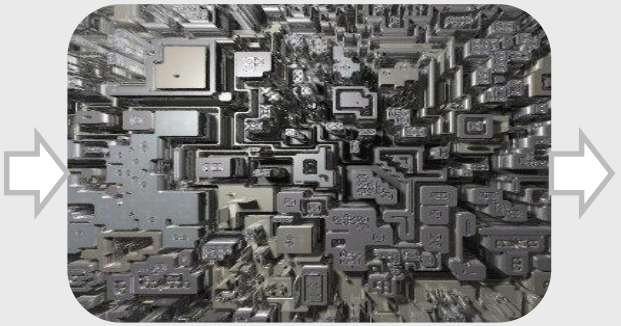
Testimise ja diagnoosi eristamine pole võimalik



Süsteemi struktureerimine teeb diagnoosi võimalikuks

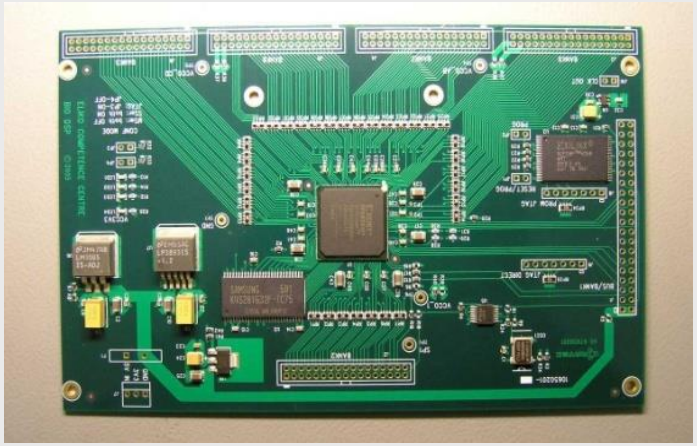


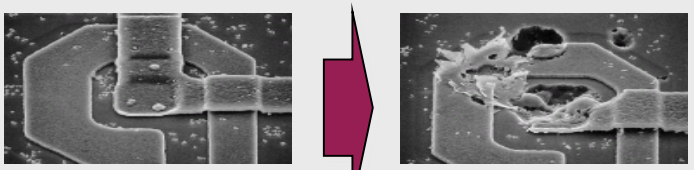
Tagasiside probleem (mälu)



Tagasisidedega seostatud suur hulk struktuurielemente toob sisse keerukuse probleemi

Hierarhiline lähenemisviis



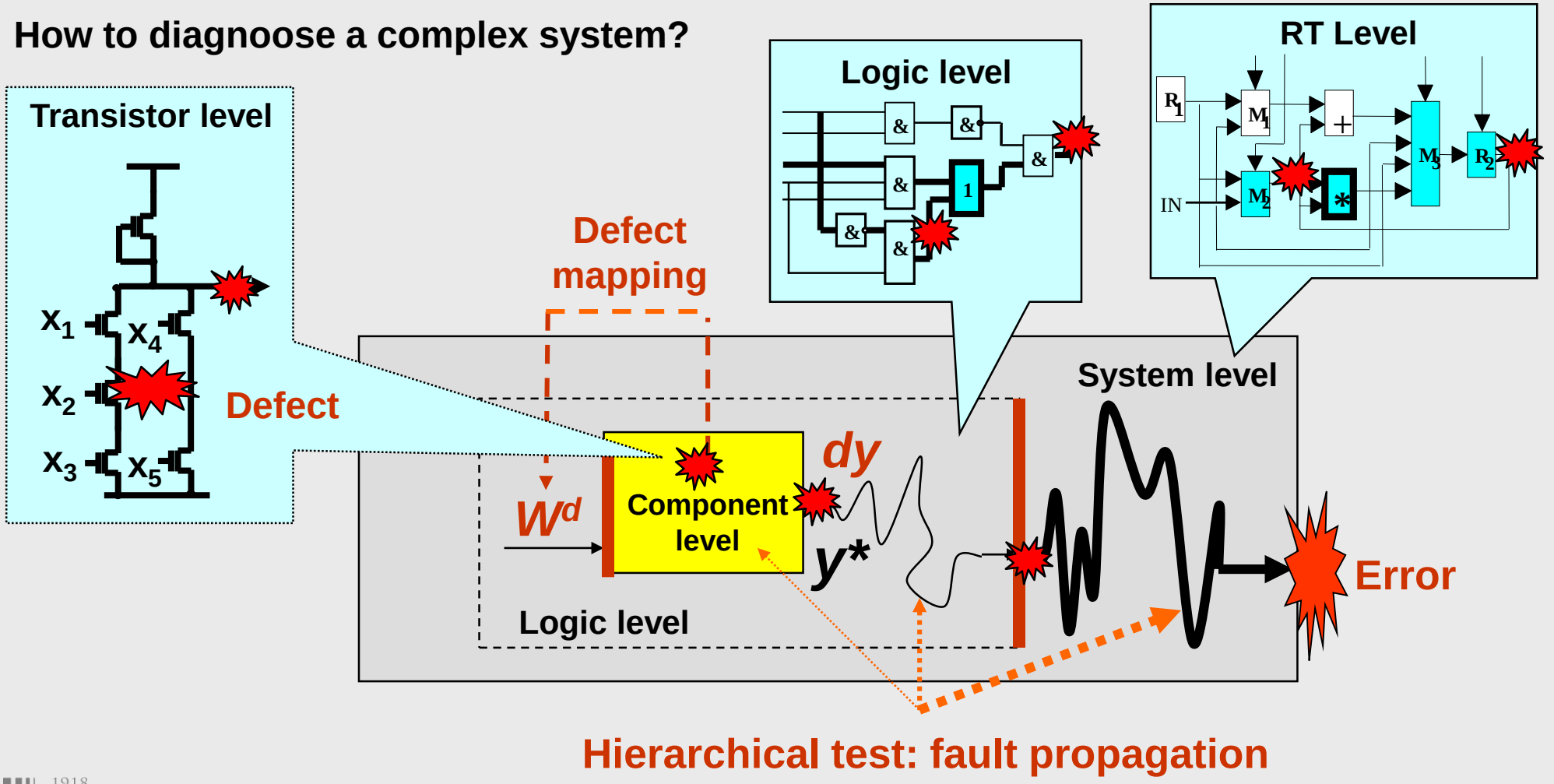


© IBM

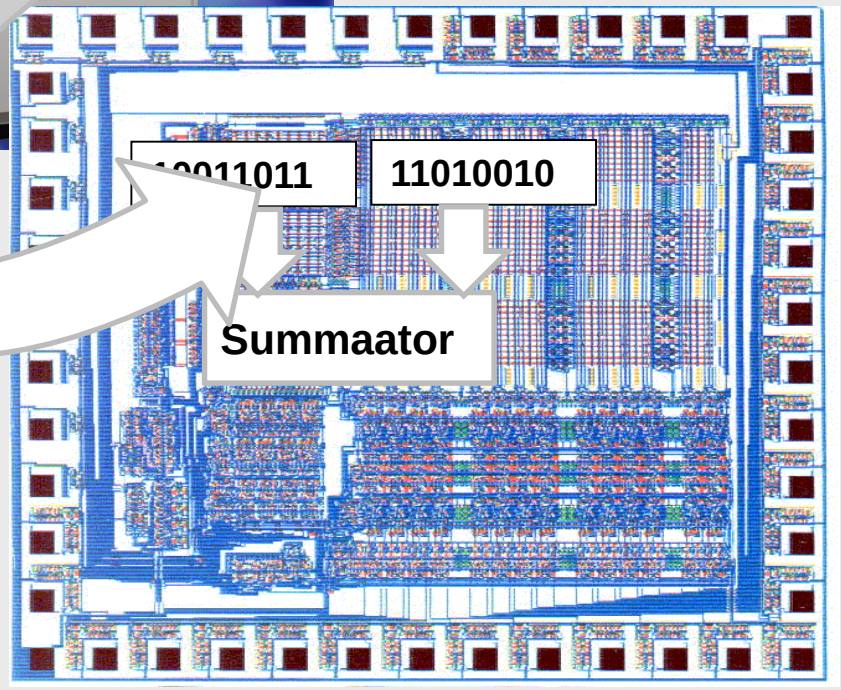
Soovitava süsteemi spetsifikatsioon

- Algoritm
- Programm
- Arhitektuur
- Struktuur
- Loogika
- Elektroonika
- Nano

Hierarchical Approach is the Solution



Testimise põhiprobleem



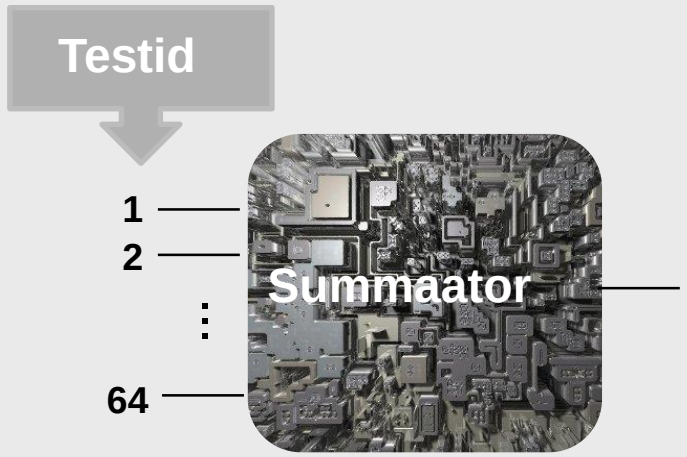
32 bitisel summaatoril on 64 sisendit
Kõigi võimalike ombinatsioonide arv on
2^{64} = 18 446 744 073 709 551 616 ≈ 10^{19}

1 GHz protsessoriga kuluks
2^{64} = 18 446 700 000 ≈ 10^{10} sek
ehk **584** aastat

Mikroprotsessori testimiseks tehase konveieril aga antakse aega vaid **10** sek

Kuidas jääb siis lugu testimise kvaliteediga?

Achilleus, kilpkonn ja digitaalsüsteemi test



Summaatori tõeväärtustabel:

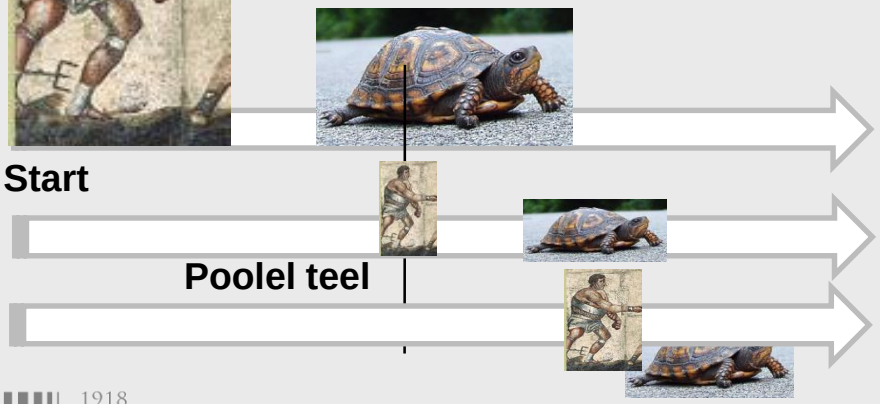
Testid	Funktsioonid
00…000	01 0 1 0 1…101
00…001	00 1 1 1 0…011
00…010	00 1 0 0 1…101
…	…
11…111	00 0 0 0 0…111

2^{64}

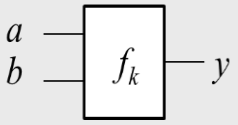
Õige tulemus

$2^{2^{64}}$

Achilleus tahtis olla ülemeelik, aga läks oma ülemeelikusega alt ja ei jõua kunagi kilpkonnale järele

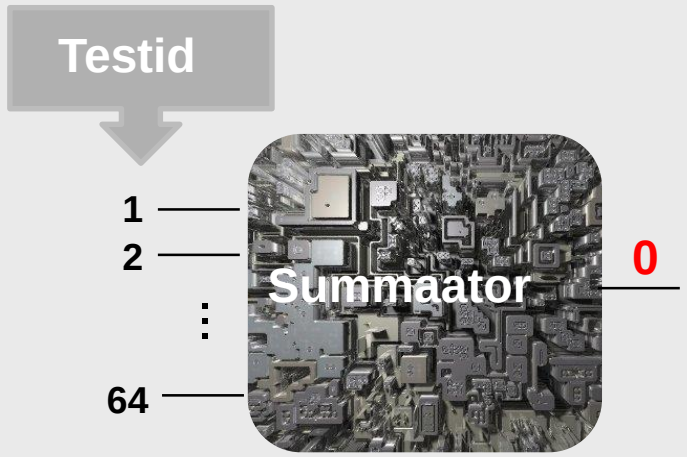


Kahe sisendiga loogikaelement võib mingi rikke tõttu realiseerida 15 võimalikku „valet“ funktsiooni



a	b	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
		"0"	&	$\overline{\rightarrow}_{ab}$	a	$\overline{\rightarrow}_{ba}$	b	$\oplus$	$\vee$	$\overline{\vee}$	~	$\overline{b}$	$\rightarrow_{ba}$	$\overline{a}$	$\rightarrow_{ab}$	$\overline{\&}$	"1"
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Achilleus, kilpkonn ja digitaalsüsteemi test



Summaatori tõeväärtustabel:

Testid	Funktsioonid
00…000	01 0 1 0 1…101
00…001	00 1 1 1 0…011
00…010	00 1 0 0 1…101
.	…
11.. 11	00 0 0 0 0…111

2^{64}

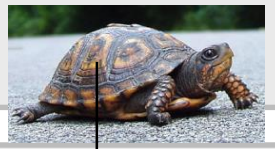
$2^{2^{64}}$

50% testitud

Esimene test

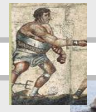
Õige tulemus

Achilleus tahtis olla ülemeelik, aga läks ise ülemeelikusega alt ja ei jõua kunagi kilpkonnale järele



Start

Poolel teel

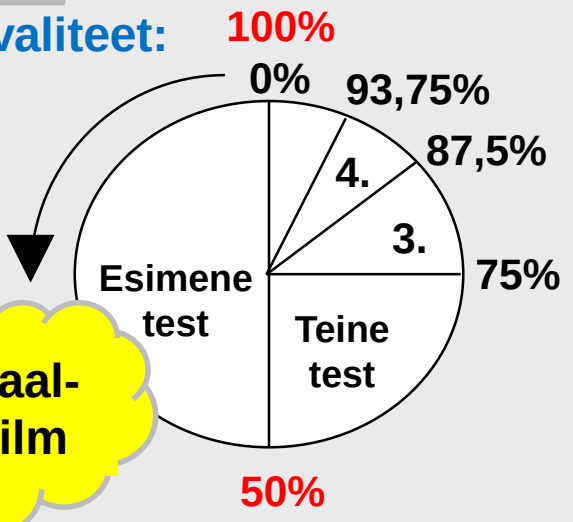


Kilpkonn ja 100%-line test on kättesaamatud mõlemad

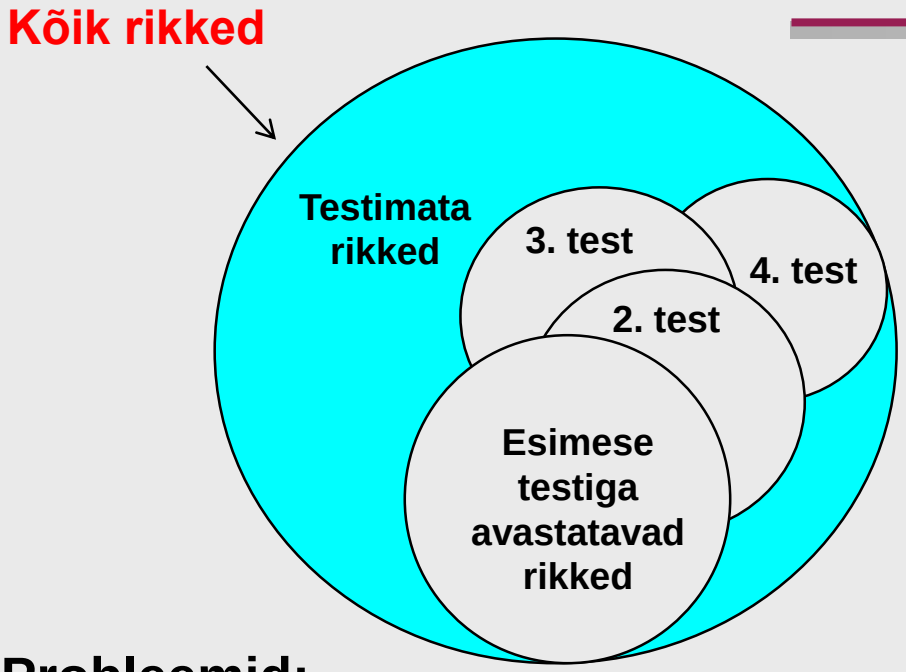
Analoog-maailm

Digitaal-maailm

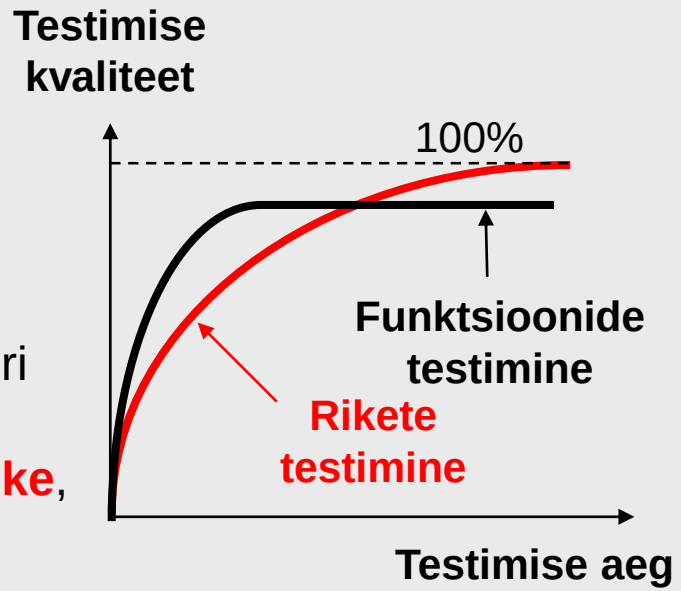
Testimise kvaliteet:



Struktuurne rikete-põhine testimine



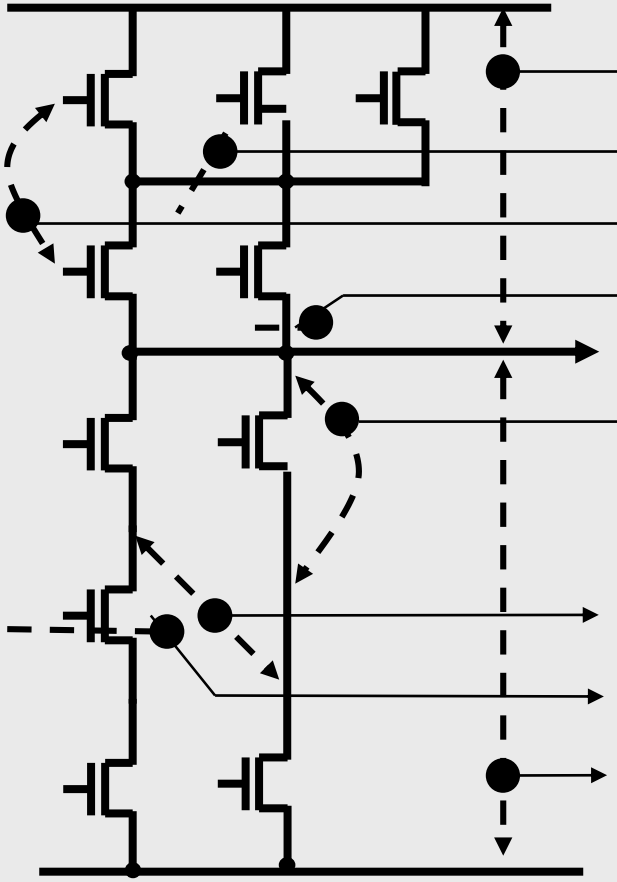
100%-line testimise kvaliteet saavutatakse, kui kõik rikked antud nimistust on kaetud



Probleemid:

- Rikete loetelu on raske koostada, eri rikkemudelite hulk on väga suur
- Rikete mudeli tööstuslik standart on **konstantrike**, kuid see on puudulik
- Kordsed rikked võivad maskeerida üksteist
- Lahendamata probleem on: **riistvara Troojad**

Rikked skeemides ja rikkemudelid?



Konstant-1

Juhe on katki

Lühis juhtmete vahel

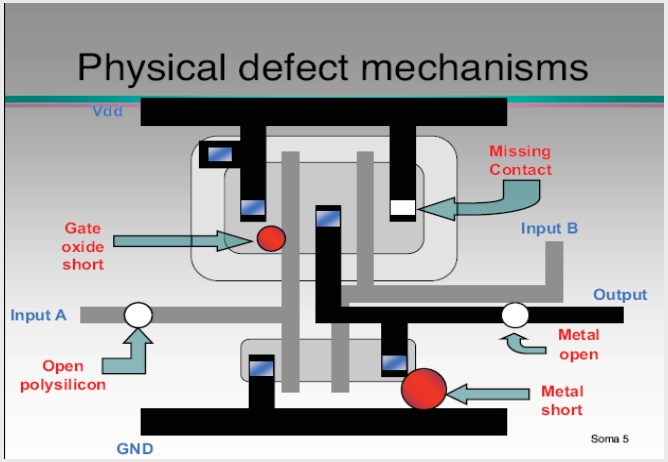
Juhe “ripub õhus”

Transistori lühis

Lühis transistoride vahel

Transistor on katki

Konstant-0

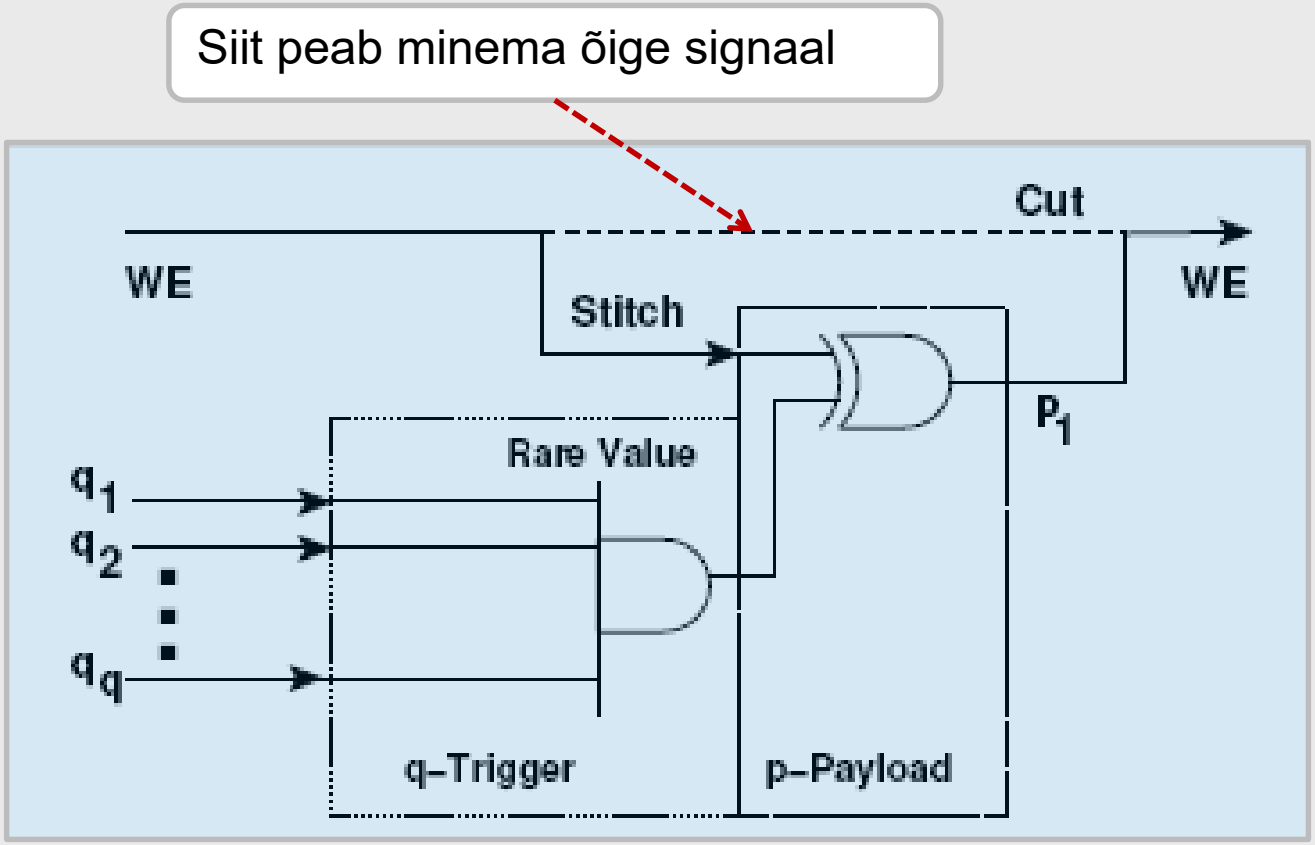


© Mani Soma

Riistvara Troojad

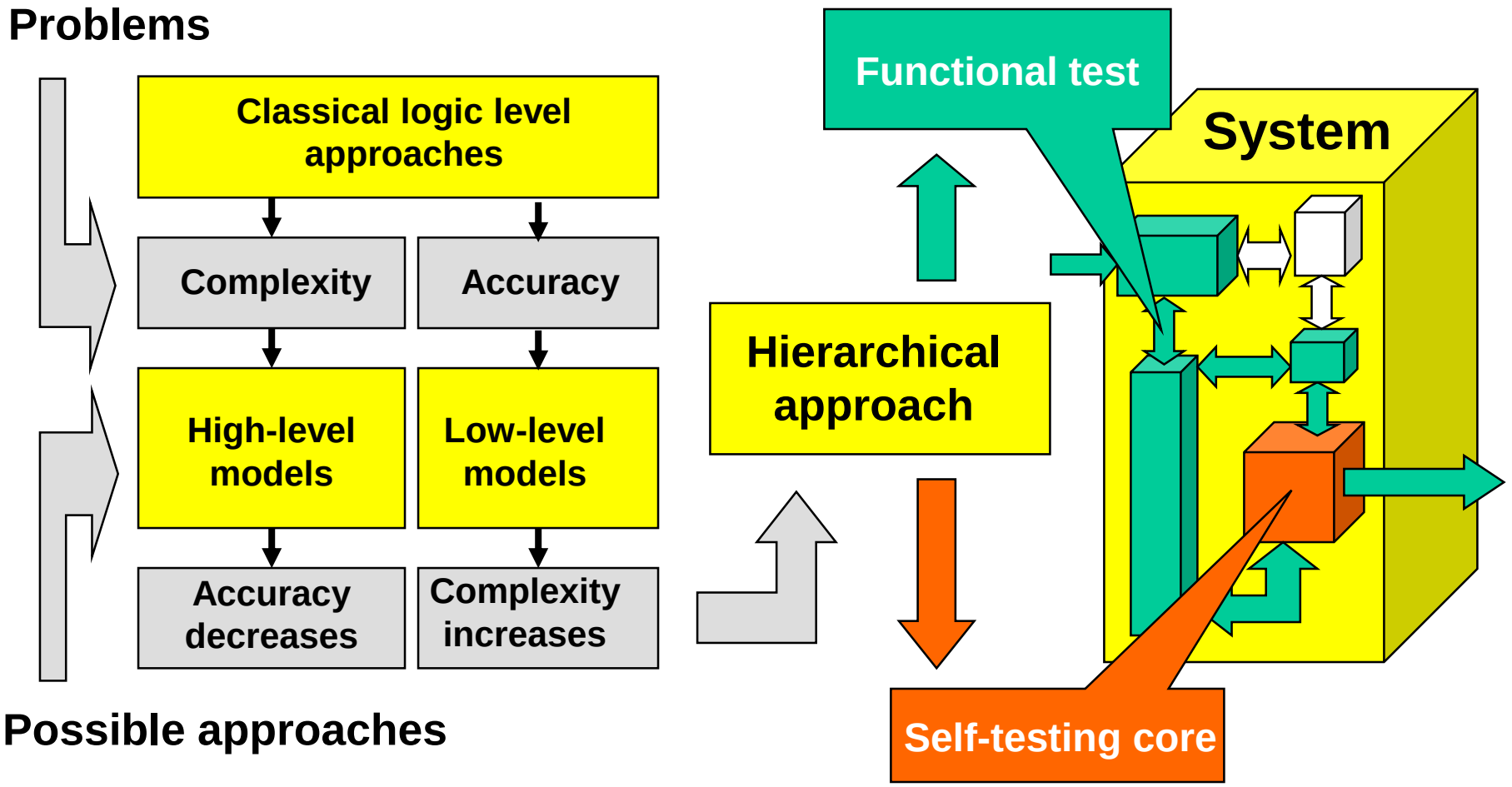
“Trooja hobune” istutatakse skeemi integraalskeemide tootmisprotsessis, kusjuures ta aktiveerub väga harva esineva oleku või mingi ajalise sündmuse puhul.

Trooja aktiveerumise korral võidakse moonutada või hävitada andmeid, edastada näiteks raadio teel salajast infot või hävitada kogu kiip.

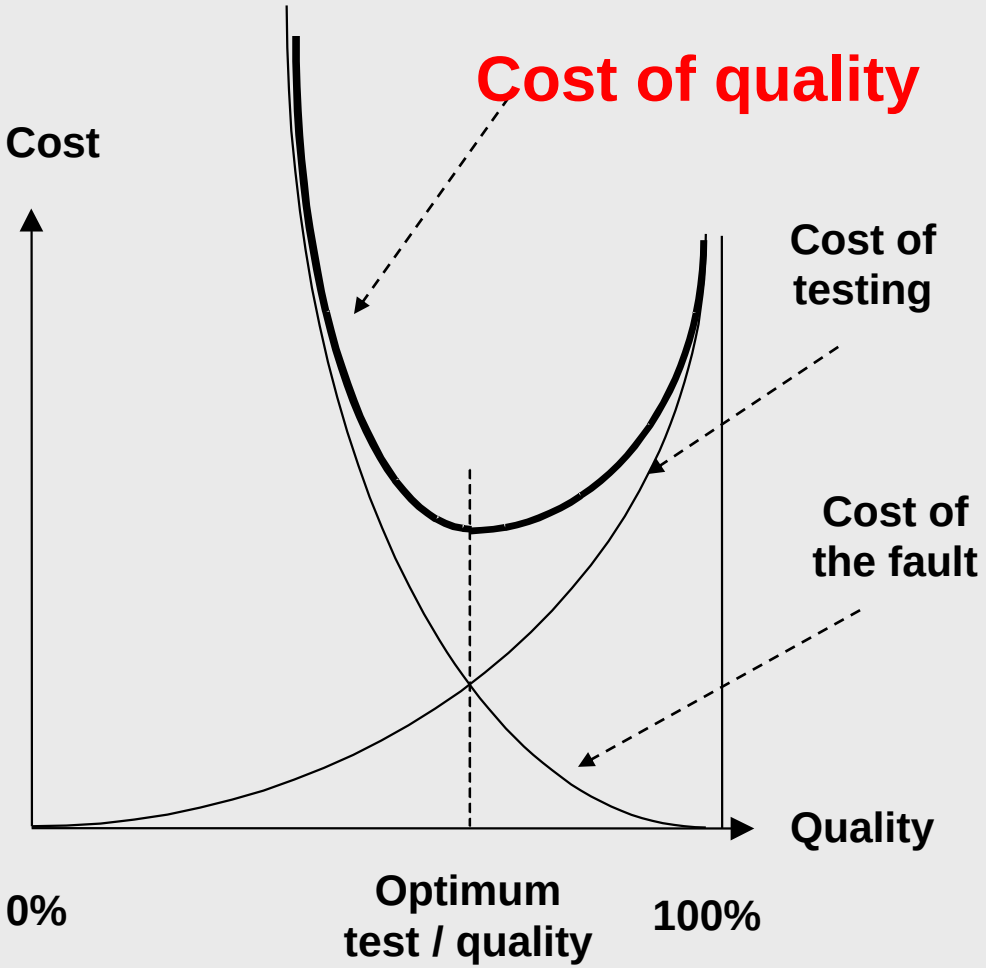


© F.Wolf, Ch. Papachristou, S.Bhunia, R.S.Chakraborty 2008

Introduction: Complexity vs. Quality

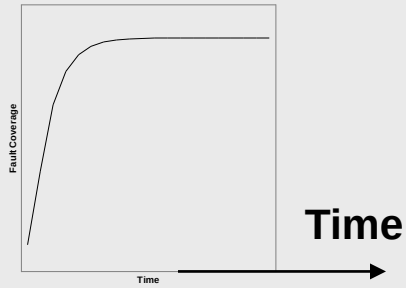


How much to test?



How to succeed?
Try too hard!
How to fail?
Try too hard!
(From American Wisdom)

Test coverage function

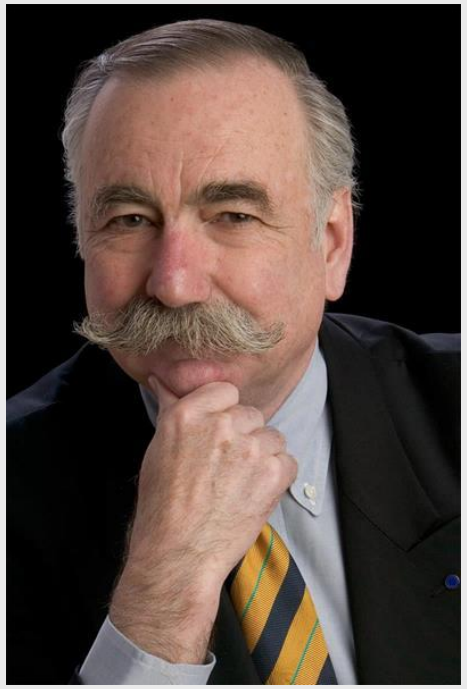


Conclusion:

"The problem of testing can only be contained not solved"

T.Williams

How much to test?



"The problem of testing can only be contained not solved"

T.Williams

Defect level - probability of selling a good product

$$DL = 1 - Y^{(1-T)}$$

Yield - probability of producing a good product

$$Y = (1 - P)^n$$

T **- test quality** (# tested faults n^*)

$n^* < n$

P **- probability of a defect**

n **- number of defects**

How Much to Test?

Amusing Test:

Paradox 1:

Digital model is finite,
analog model is infinite.

However, the complexity problem
was introduced
by Digital World

Paradox 2:

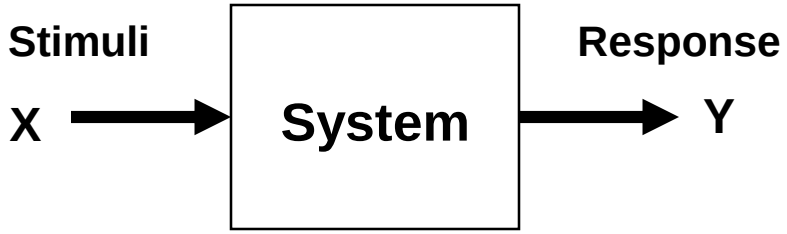
If I can show that the system works,
then it should be not faulty.

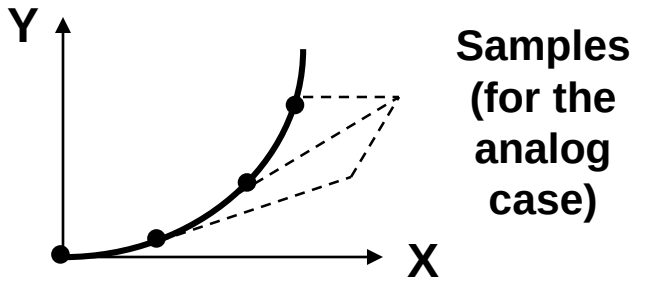
But, what does it mean: it works?

32-bit accumulator has 2^{64} functions
which all should work.
So, you should test all of them!

All life is an experiment.
The more experiments you make,
the better

(American Wisdom)





In digital case you cannot
extrapolate

How Much to Test?

Paradox 3:

2^{64} input patterns (!)
for 32-bit accumulator
will be not enough.

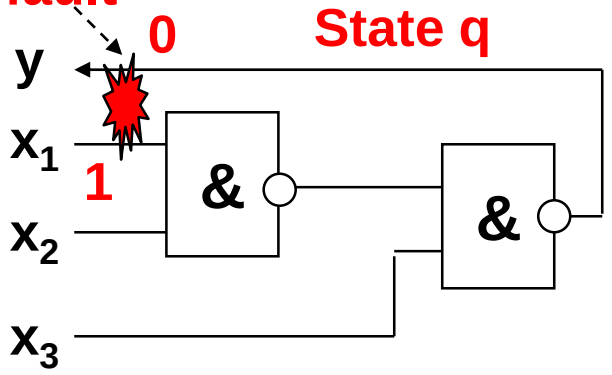
A short will change the circuit
into sequential one,
and you will need because of that

2^{65} input patterns

> Time can be your best friend
> or your worst enemy
> (Ray Charles)

$Y = F(x_1, x_2, x_3)$

Bridging fault



$Y = F(x_1, x_2, x_3, q)$

Paradox 4:

Mathematicians counted that Intel 8080
needed for exhaustive testing 37 (!) years

Manufacturer did it by 10 seconds
Majority of functions will never activated
during the lifetime of the system

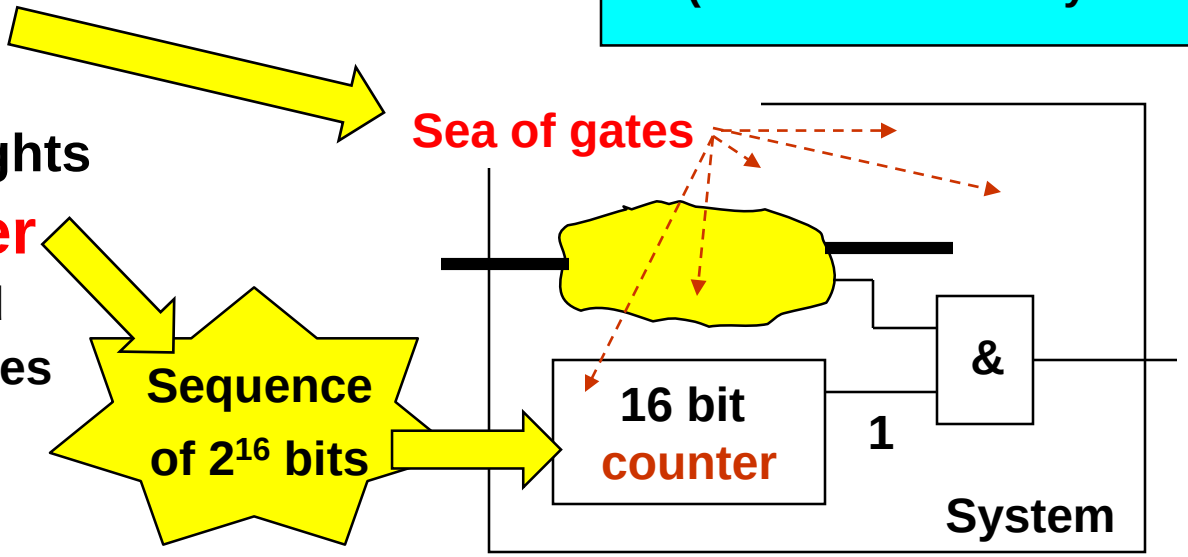
Introduction: Hierarchy is Important

Paradox 5:

To generate a test
for a block in a system,
the computer
needed
2 days and 2 nights

An engineer
did it by hand
with 15 minutes

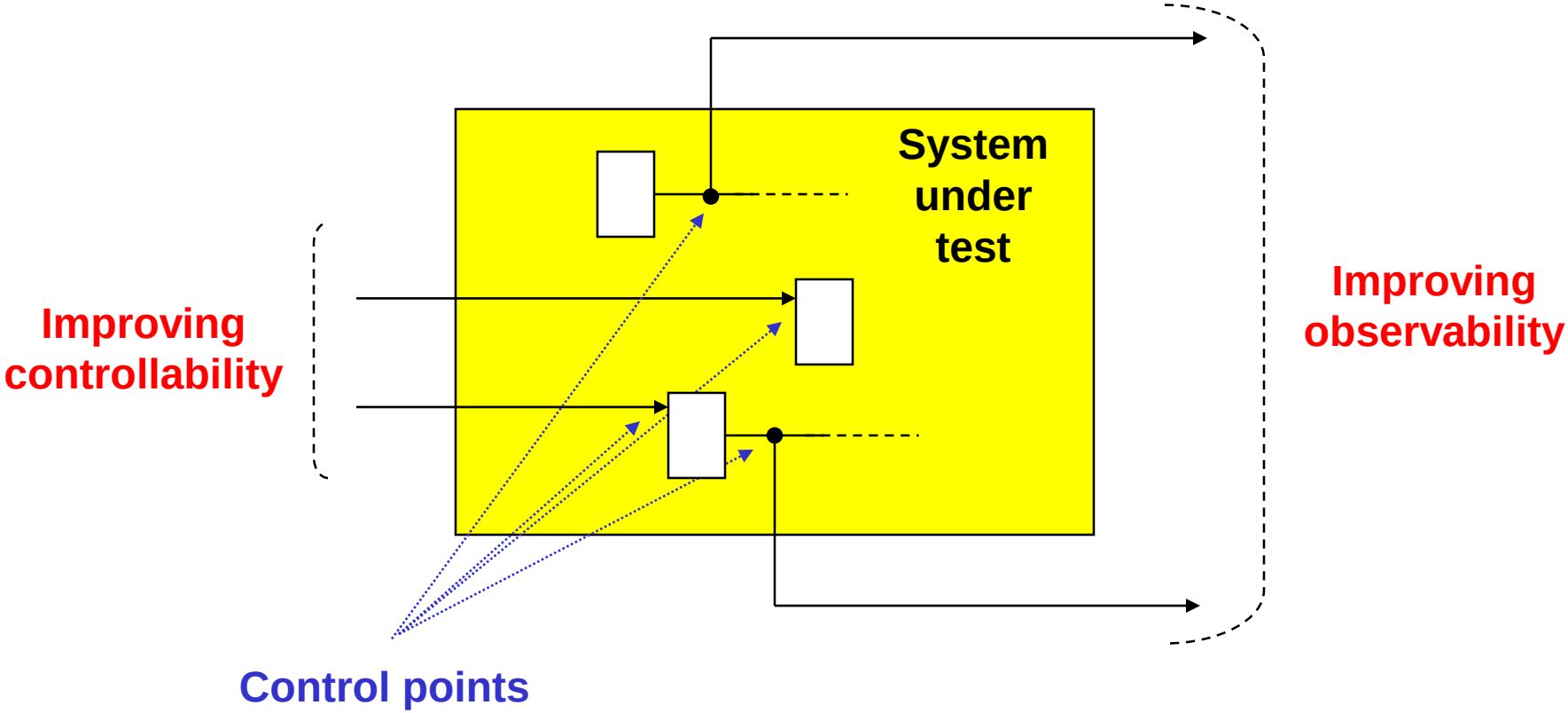
The best place to start is with a good title. Then build a song around it.
(Wisdom of country music)



A call for Design for Testability

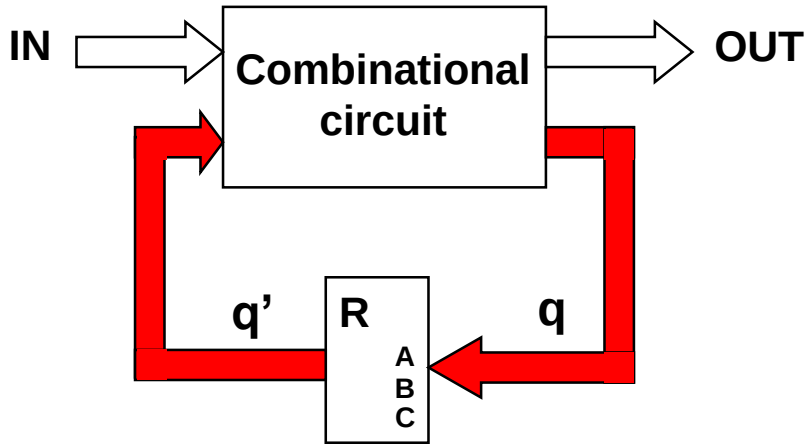
Design for Testability

Improving testability with inserting of control points

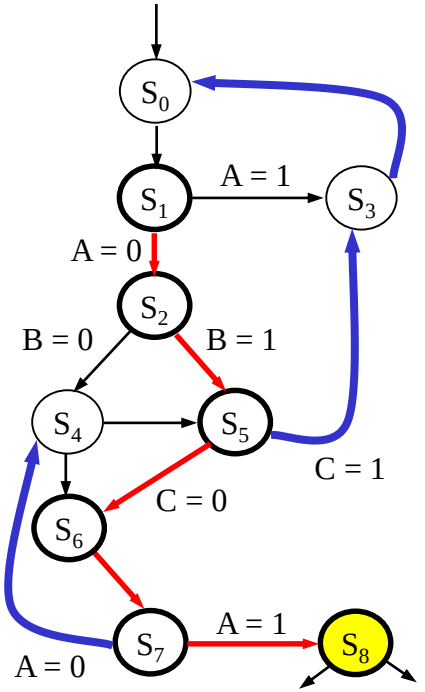


Design for Testability

theisleofwightcomputergeek.co.uk



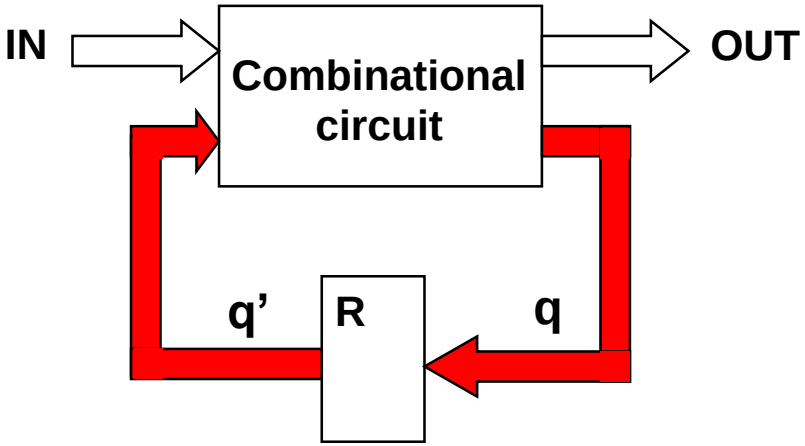
Why are sequential circuits difficult to test?



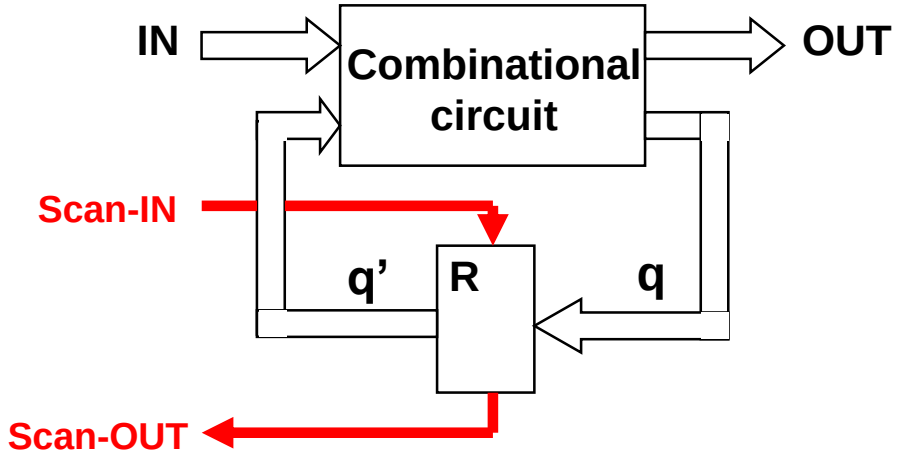
Due to the feedback loop it is difficult to generate an input sequence which brings the circuit into a desired state

Design for Testability

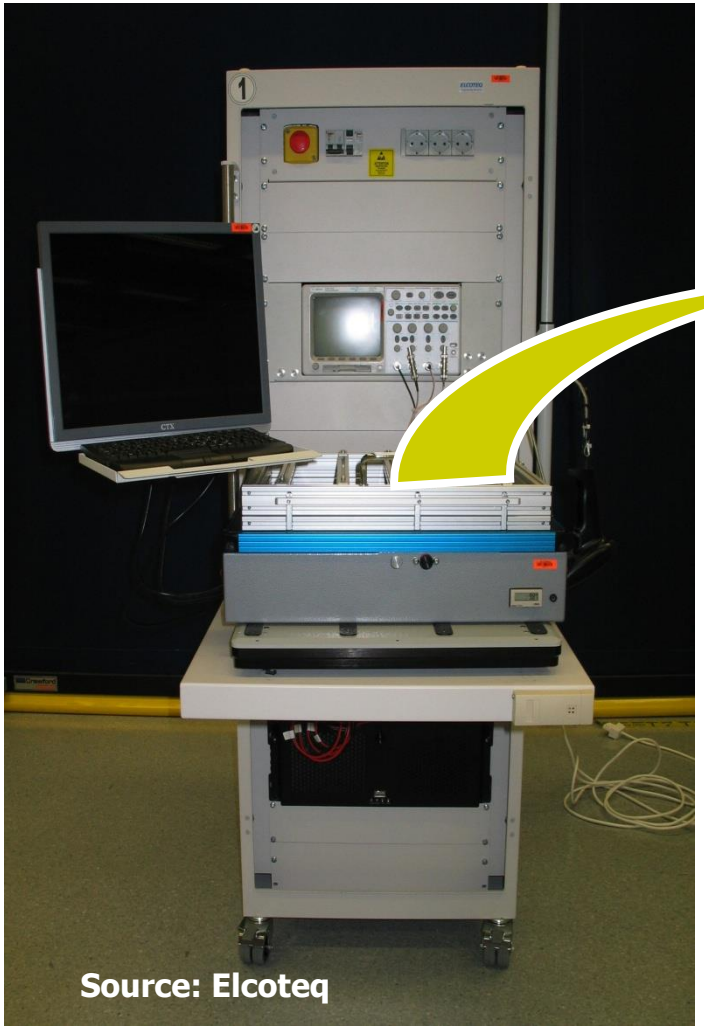
theisleofwightcomputergeek.co.uk



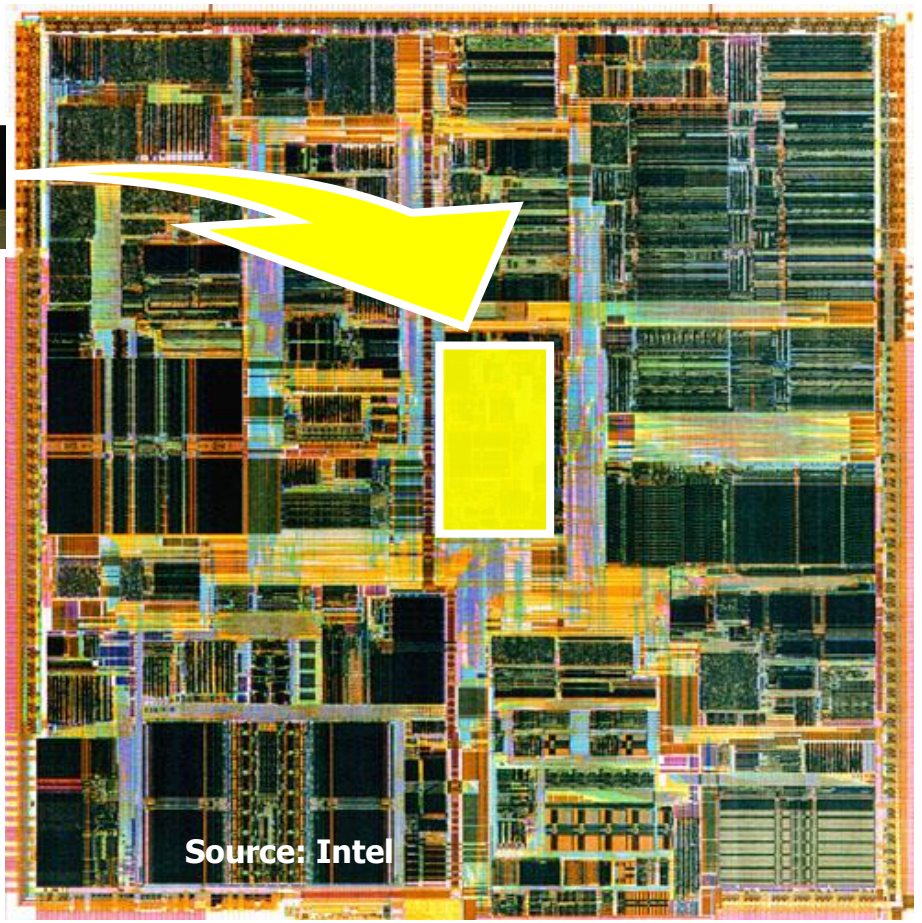
Scan-Path design strategy



Testing Challenges: Built-in Self-Test



Cores have to be tested on chip

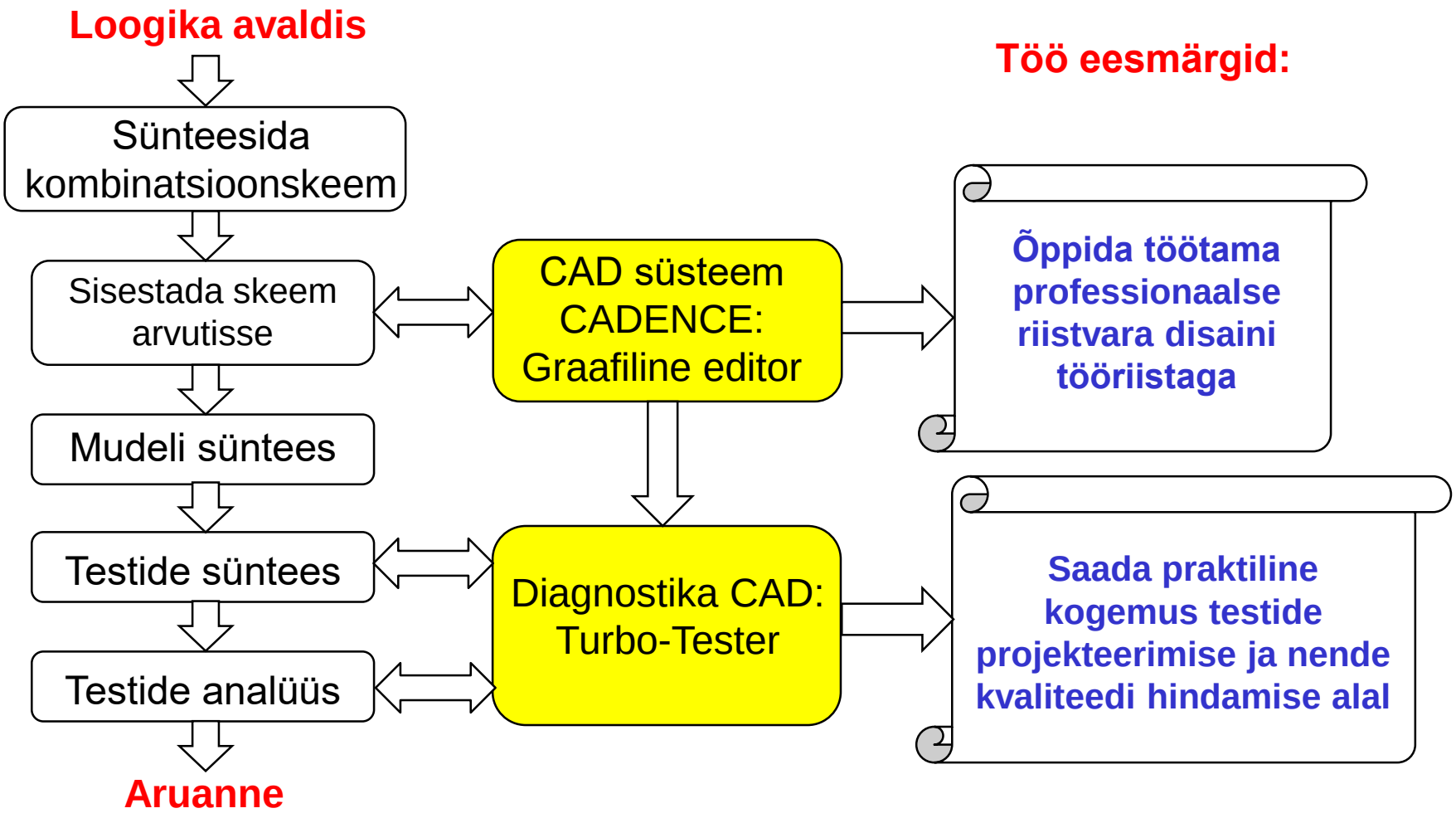


Süsteemide diagnostika

1. Sissejuhatus

- 1.1. Kursuse eesmärgid ja probleemi püstitus
- 1.2. Kursuse sisu ja struktuur
- 1.3. Süsteemide diagnostika põhiprobleemid
- 1.4. Kursusetöö ja bakalaureusetöö temaatikast

Iseseisev ülesanne (kursusetöö)



Iseseisev ülesanne (kursusetöö)

- Õppida tundma
  - digitaalskeemide rikete mudeleid,
  - testide sünteesi ja analüüsi meetodeid ning
  - uurida eksperimentaalselt nende kasutamise efektiivsust digitaalskeemides
- Kasutada vastavaid CAD vahendeid
  - skeemide sisestamiseks arvutisse ja
  - diagnostikaprobleemide lahendamiseks

Iseseisev ülesanne (kursusetöö)

1) Uurimisobjektina sünteesida kombinatsioonskeem Boole'i funktsioonile Y = $y_i \vee y_j$
   - Kasutada avaldistes ainult loogikaelemente AND, OR ja NOT
   - Mitte muuta seejuures avaldistega etteantud loogikastruktuuri, s.t. mitte kasutada lihtsustusi ja optimeerimist

$$y_i = \overline{x_{11}}(x_{21} \vee x_{31})(x_{41} \vee \overline{x_{22}}) \vee x_{12}(\overline{x_{51}}x_{61} \vee x_{23}) \vee x_{52}\overline{x_{62}} \vee \overline{x_{24}}\,\overline{x_{32}}x_{42}$$

$$y_j = x_{21}(x_{51} \vee \overline{x_{11}}x_{31}) \vee x_{61} \vee \overline{x_{22}}(x_{52}\overline{x_{62}} \vee x_{32}) \vee (x_4 \vee \overline{x_{12}}x_{53}\overline{x_{63}})$$

   - Väärtused (i,j) on antud tabelis vastavalt variandi numbrile:

Iseseisev ülesanne (kursusetöö)

2) Sisestada skeem arvutisse, kasutades CAD süsteemi Cadence redaktorit
3) Lahendada kaks Boole'i differentsiaalvõrrandit (k=1, ... 6):

$$\frac{\partial Y}{\partial y_i}\frac{\partial y_i}{\partial x_{k1}} = 1 \qquad \frac{\partial Y}{\partial y_j}\frac{\partial y_j}{\partial x_{k1}} = 1$$

- Kus Variandi numbri N puhul : k = N(mod 6)
- NB! Indeks k1 tähendab, et tegemist on literaaliga (avaldises vasakult poolt esimesega), mitte muutujaga x_k
- Siin on mõeldud, et x_{k1} muutub, aga x_k jääb ise muutumatuks (s.t. rike on skeemi harus x_{k1}, aga mitte sisendis x_k**)**

Iseseisev ülesanne (kursusetöö)

4) Sünteesida projekteeritud skeemi struktuurne otsustusdiagramm.
5) Sünteesida projekteeritud skeemi funktsionaalne optimeeritud otsustusdiagramm. Konstrueerida diagrammi abil testid kõigile kuuele sisendile.
6) Genereerida käsitsi vabal meetodil testid, mis avastaksid projekteeritud skeemis kõik mitteliiased rikked const. 0 ja const 1. Teha kindlaks, millised rikked on liiased.

6a) Koostada rikete tabel. Genereerida optimeeritud diagnostikaprogramm

7) Genereerida käsitsi vabal meetodil testid, mis avastaksid projekteeritud skeemis lühised järgmiste sisendite vahel:

   1/2, 3/4, 5/6, 2/3, 4/5

   Kasutada lühise AND-mudelit

Iseseisev ülesanne (kursusetöö)

8) Genereerida testid, mis avastaksid viitevead haruteedel

   x_{11}, x_{21}, x_{31}, x_{41}, x_{51} ja x_{61}

9) Leida, milliseid konstant tüüpi rikkeid avastavad testvektorid:

   000000, 111111, 010101, 101010, 001100, 110011, 000111 ja 111000.

10) Määrata punktis 6 projekteeritud testide kvaliteet diagnostikatarkvara Turbo-Tester (TT) rikete simulaatoriga. Märkida ventiilskeemile avastamata jäänud rikked. Trükkida välja rikete diagnostikatabel.

11) Määrata rikete tabeli abil skeemi diagnoosid juhtumitel, kus punktis 6 projekteeritud testi puhul järgmised vektorid annavad negatiivse tulemuse (s.t. avastavad vea): (1,2), (2,4,5) ja (3,6).

Iseseisev ülesanne (kursusetöö)

12) Genereerida TT-ga skeemile test. Võrrelda TT testi ja käsitsi projekteeritud testide kvaliteete

13) Võrrelda TT deterministliku ja juhuslike arvude generaatoril põhineva testide sünteesi effektiivsusi ja erinevusi nii ventiil- kui ka makrotasemel esitatud objektide puhul.

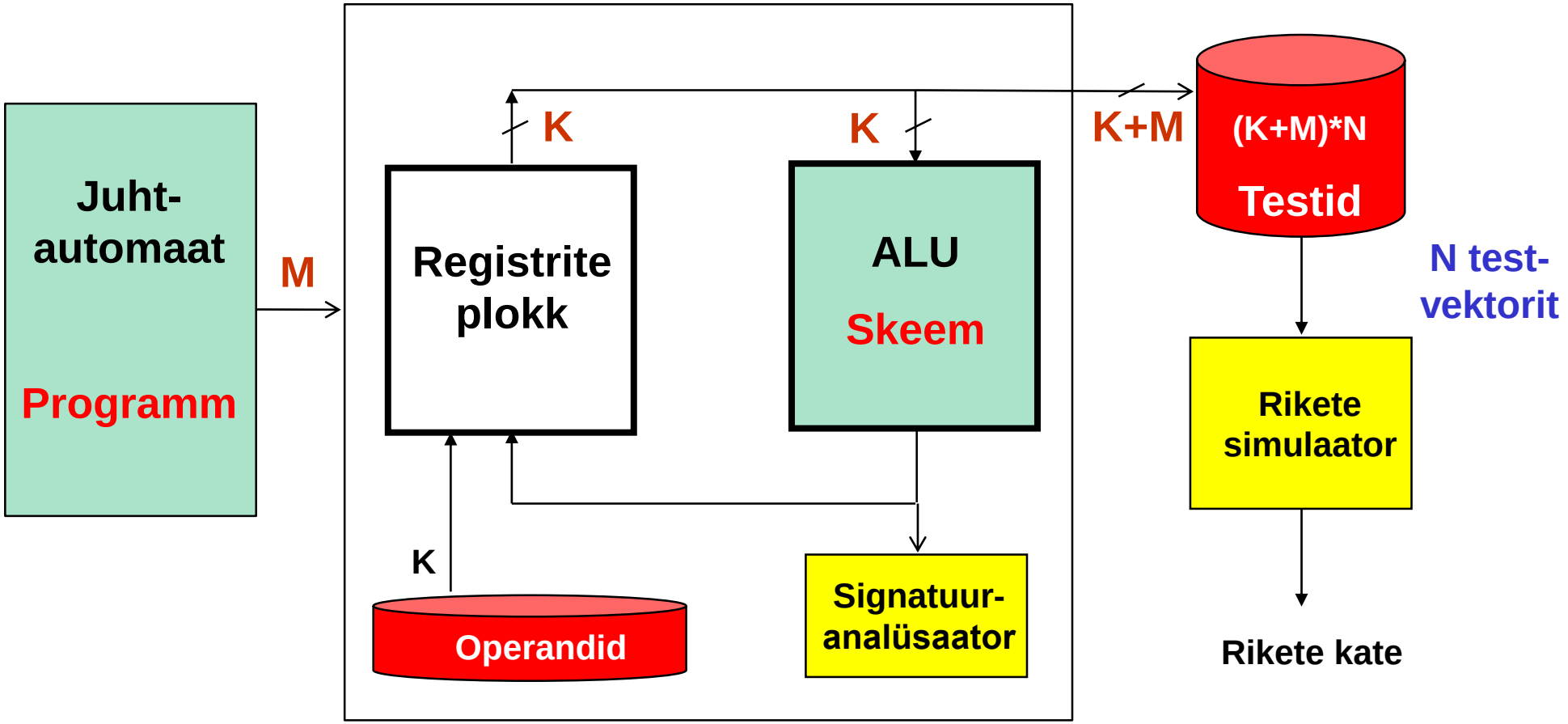
    Miks on rikete arvud nendel mudelitel erinevad?

    Miks on testide sünteesi efektiivsus erinev?

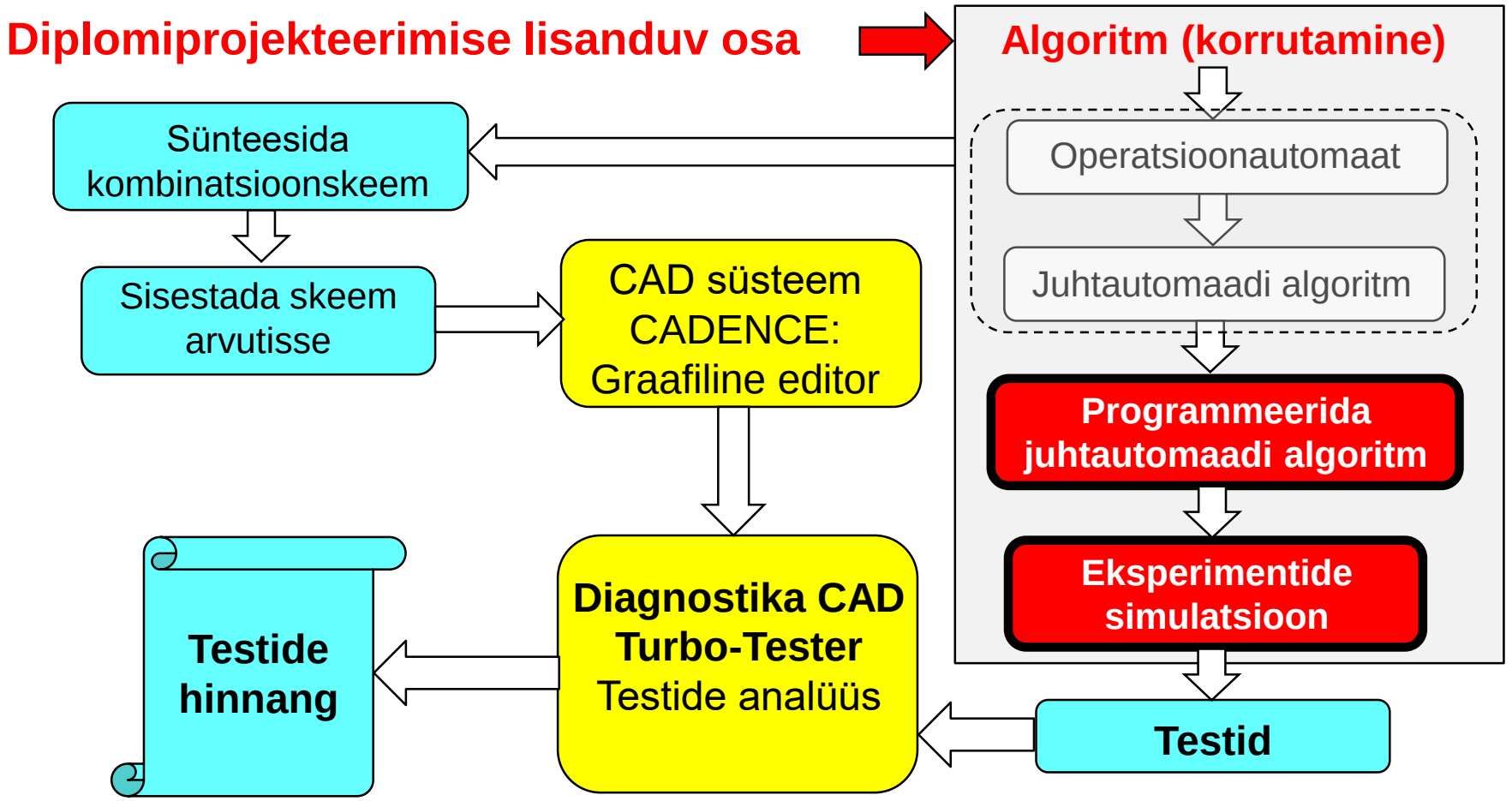
14) Esitada aruanne, mis sisaldab:
    - projekteeritud skeemi väljatrükk
    - skeemi mudelid struktuurse ja optimeeritud funktsionaalse otsustusdiagrammina
    - vastused kõikidele ülal esitatud küsimustele ja ülesannetele
    - kokkuvõte ja järeldused tehtud tööst

Bakalaureusetöö temaatikatest

Operatsioonautomaadi töö monitoorimine programmi abil:

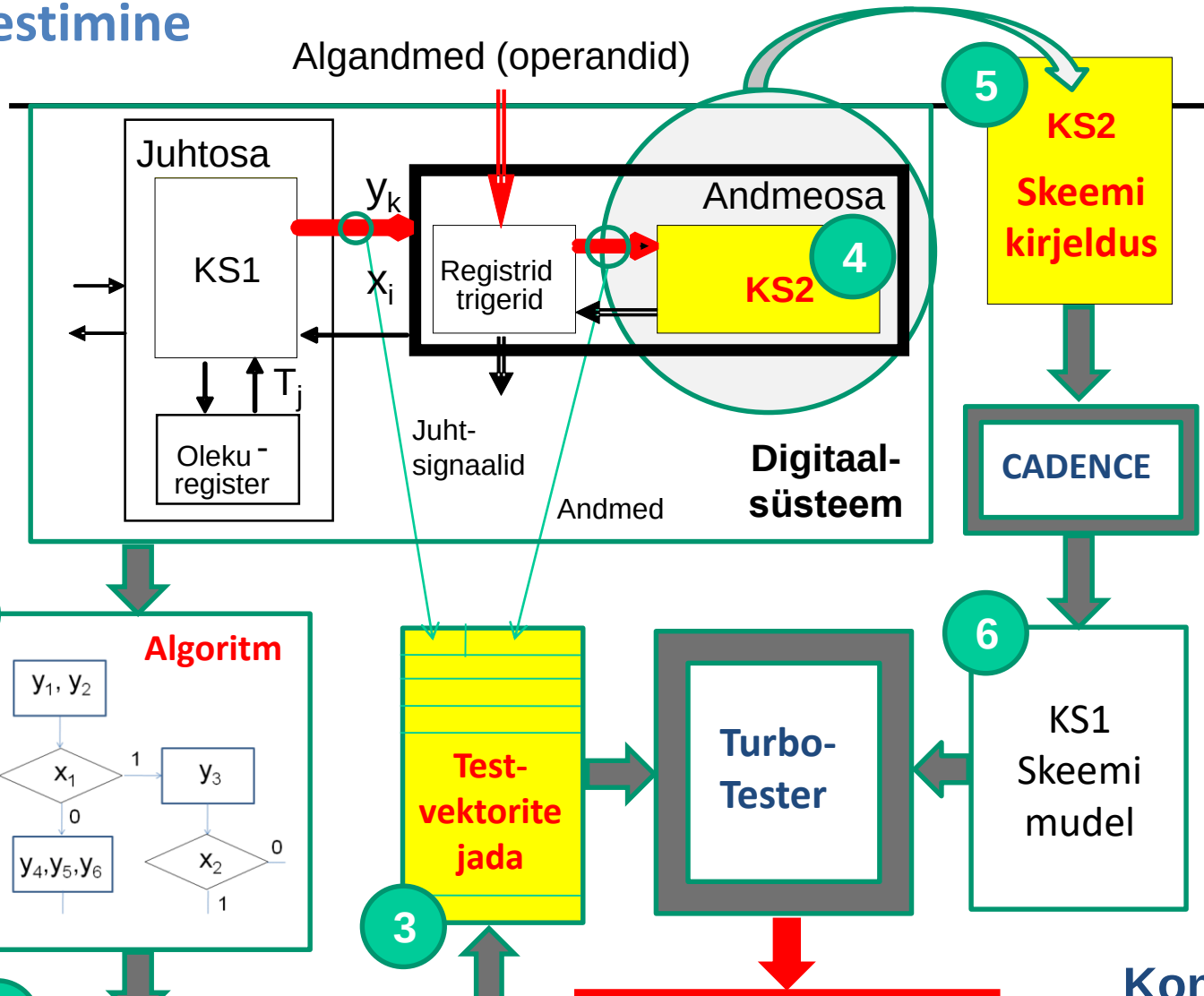


Bakalaureusetöö temaatikatest



Bakalaureusetöö teema

Digitaalsüsteemi funktsionaalne testimine



Töö sisu:

1) Konstrueerida ette antud aritmeetikatehet realiseeriv **algoritm**
2) Koostada algoritmi imiteeriv **programm**
3) Leida programmiga vahetulemuste (x_i,T_j) jada ehk on-line funktsionaalne **test** ploki KS1 jaoks
4) Projekteerida testitav **skeem KS2**
5) Sisestada skeem CADENCE redaktori abil arvutisse
6) Simuleerida skeem leitud testjadal Turbo-Testri rikete simulaatoriga ning **määrata testi kvaliteet**

Kontakt: **prof. R.Ubar**
raiub@pld.ttu.ee