1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Testimise projekteerimine:

Labor 1
Understanding BIST

Sergei Kostin

skostin@ati.ttu.ee

Laborid

BISTA

1. **Understanding BIST (6. nädal)**
2. **BIST Optimization (10. nädal)**

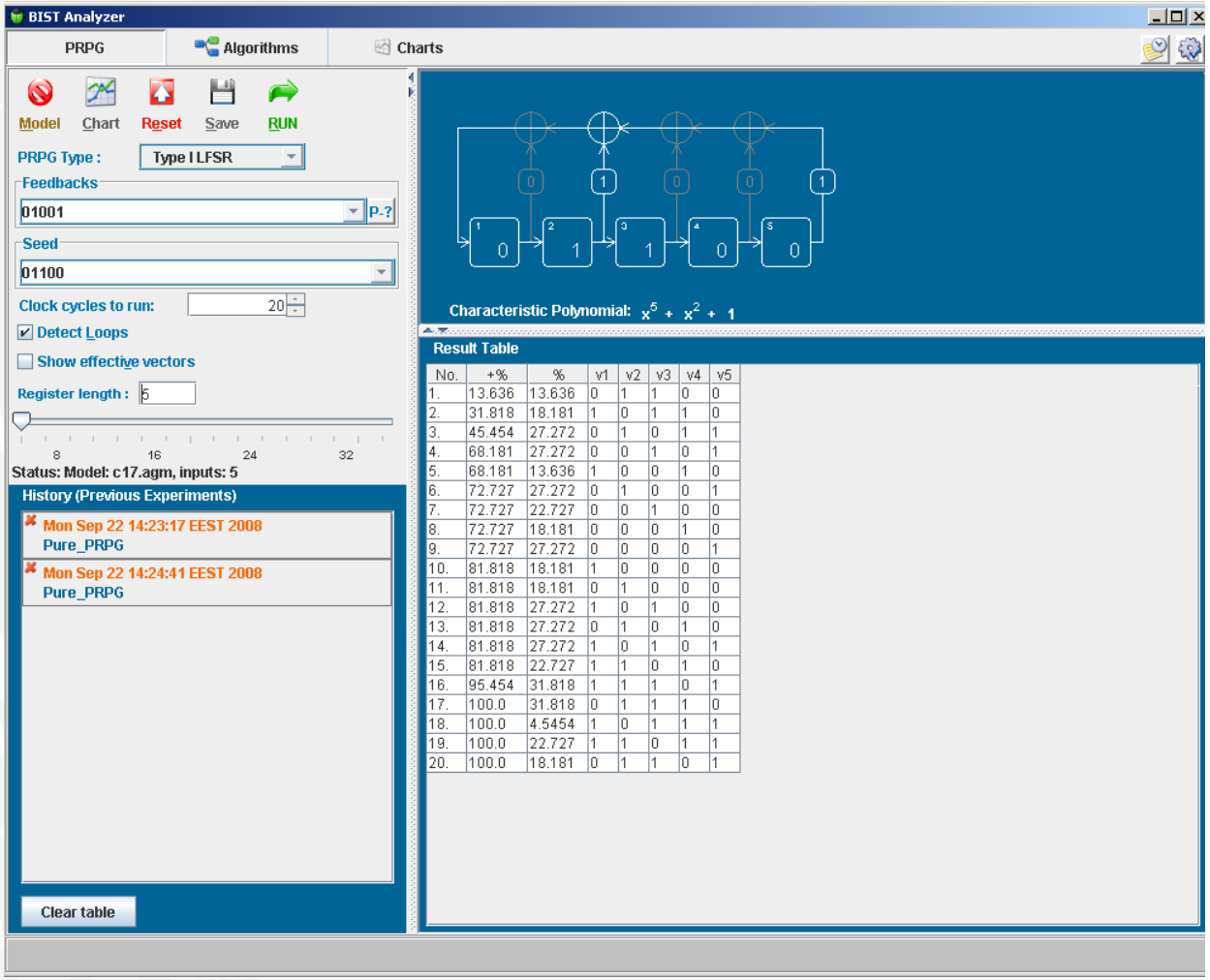
Trainer 1149

3. **Understanding Boundary Scan (14. nädal)**

Cadence BISTA Turbotester

Kursuse töö (eksam)

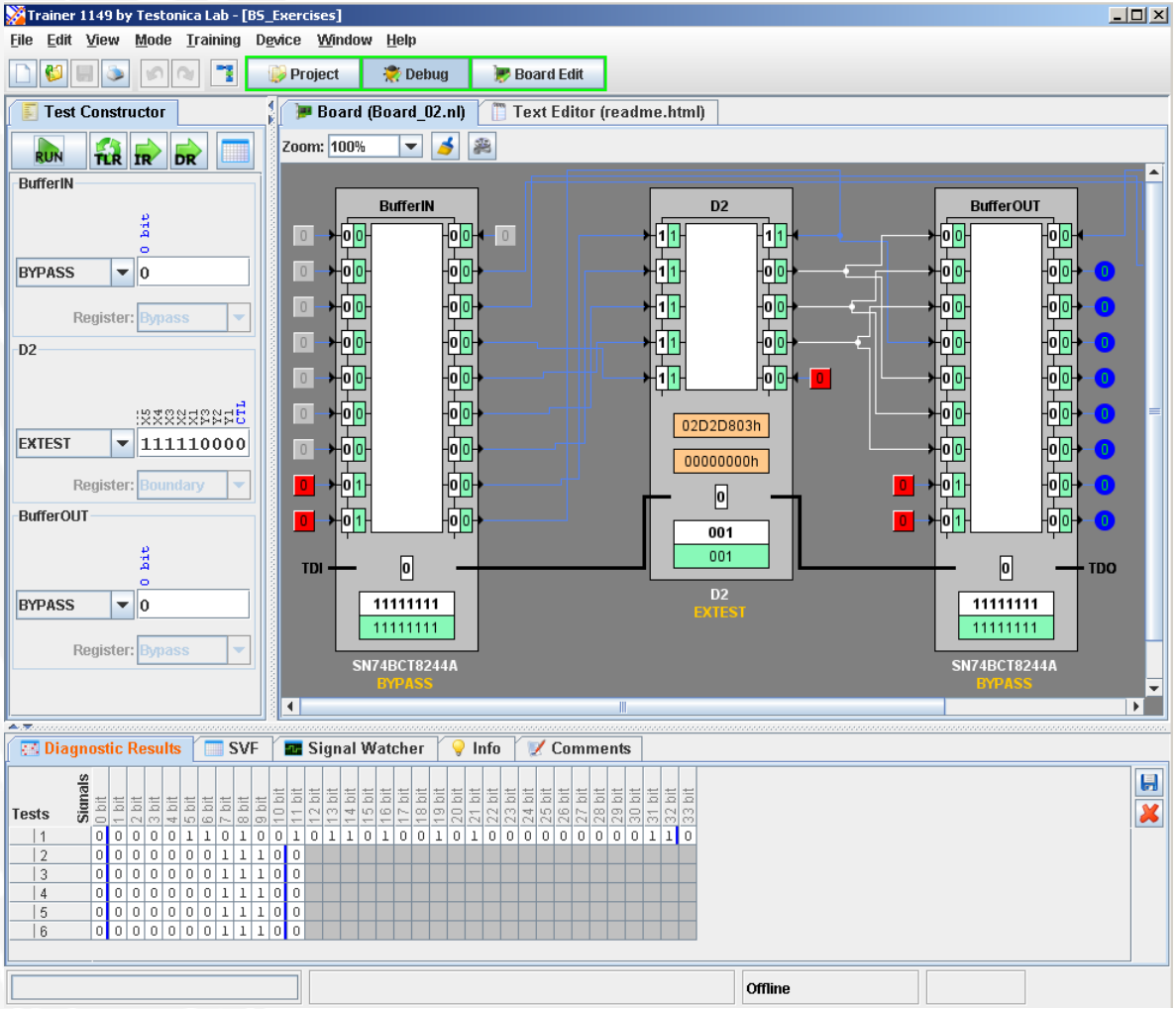
BISTA (Built-in Self Test Analyser)
BIST Technique Teaching Tool



No.	+%	%	v1	v2	v3	v4	v5
1.	13.636	13.636	0	1	1	0	0
2.	31.818	18.181	1	0	1	1	0
3.	45.454	27.272	0	1	0	1	1
4.	68.181	27.272	0	0	1	0	1
5.	68.181	13.636	1	0	0	1	0
6.	72.727	27.272	0	1	0	0	1
7.	72.727	22.727	0	0	1	0	0
8.	72.727	18.181	0	0	0	1	0
9.	72.727	27.272	0	0	0	0	1
10.	81.818	18.181	1	0	0	0	0
11.	81.818	18.181	0	1	0	0	0
12.	81.818	27.272	1	0	1	0	0
13.	81.818	27.272	0	1	0	1	0
14.	81.818	27.272	1	0	1	0	1
15.	81.818	22.727	1	1	0	1	0
16.	95.454	31.818	1	1	1	0	1
17.	100.0	31.818	0	1	1	1	0
18.	100.0	4.5454	1	0	1	1	1
19.	100.0	22.727	1	1	0	1	1
20.	100.0	18.181	0	1	1	0	1

Trainer 1149

Boundary Scan Technique Teaching Tool



Uus klassi kasutajate süsteem

http://elrond.tud.ttu.ee/

Kuidas saada kasutajakontot:

•**Et saada klassi kasutajakontot pead ennast juba eelnevalt olema registreerinud meie LDAP baasis. Kasutajakonto loomine toimub üle ÕIS-i (ois.va.ttu.ee → TTÜ üldparooli loomine).**

•**Järgmiseks etapiks on klassi serverisse kasutajakonto loomine. Selleks pöördu webi lehele auth.ttu.ee, logi sisse oma kasutajanime ning parooliga ning kliki lingile "loo klassi kasutaja konto". Kindluse mõttes küsitakse veel korra parooli, mis langeb kokku sisselogimise parooliga ning genereeritakse kasutaja loomise päring. Päringuid töödeldakse perioodiliselt kuid MITTE KOHE pärast päringu sisestamist. Päringu täitmise staatust saab kontrollida webi lehelt auth.ttu.ee.**

Juhendid arvutite seadmiseks

Esmasel sisselogimisel järgnevad sammud:

1. **vahetada oma shell ümber, selleks anda käsklus konsoolis ‘passwd -r ldap -e’**
2. **küsitakse parooli ja uue shelli väärtust, uueks shelliks panna: ‘/bin/tcsh’**
3. **.cshrc puudumisel kopeerida endale õige, järgneva käsuga ‘cp /home/kasutaja/.cshrc .’**
4. **Võta maha kommentaarid(e. #) .cshrc failis ridadel:**

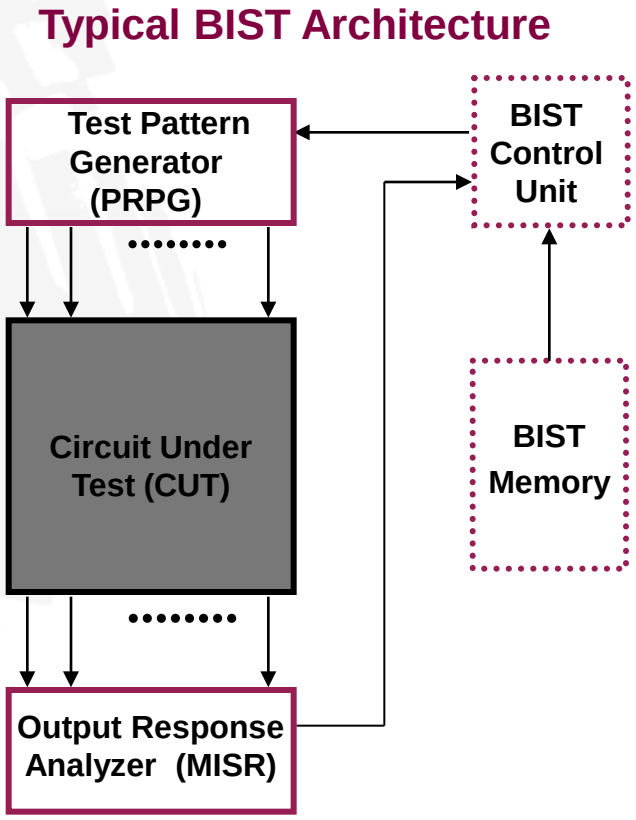
   setenv TESTER

   setenv JDK16

5. **logida end arvutist välja ning oodata 60 sekundit enne uut sisselogimist**

BIST (Built-in Self Test)

ehk *sisseehitatud isetestimine* on digitaalskeemi (mikroskeemi, plaadi, süsteemi jms) omadus iseennast testida.



Why BIST?

Motivations for BIST:

- Need for a cost-efficient testing (general motivation)
- Doubts about the stuck-at fault model
- Increasing difficulties with TPG (Test Pattern Generation)
- Growing volume of test pattern data
- Cost of ATE (Automatic Test Equipment)
- Test application time
- Gap between tester and UUT (Unit Under Test) speeds

Drawbacks of BIST:

- Additional pins and silicon area needed
- Decreased reliability due to increased silicon area
- Performance impact due to additional circuitry
- Additional design time and cost

Test Patterns Generator

- Store in ROM – too expensive
- Exhaustive
- Pseudo-exhaustive
- Pseudo-random (LFSR) – Preferred method
- Binary counters – use more hardware than LFSR
- Modified counters
- Test pattern augmentation
  - LFSR combined with a few patterns in ROM
  - Hardware diffracter – generates pattern cluster in neighborhood of pattern stored in ROM

BIST: Exhaustive test

Universal test sets

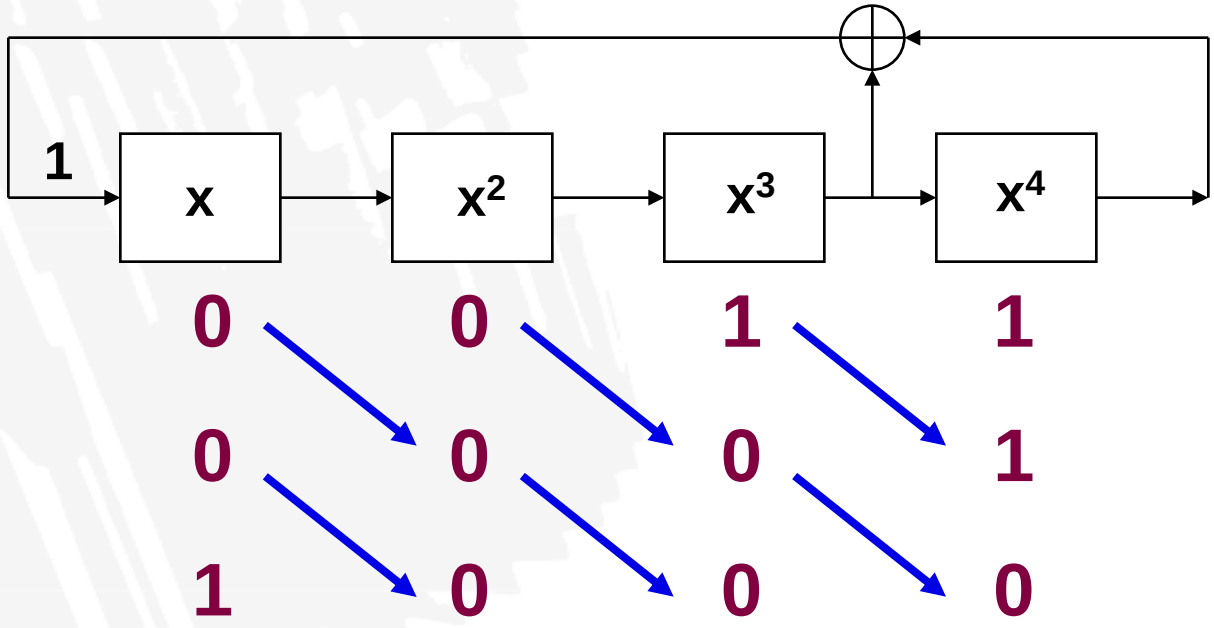
1. **Exhaustive test (trivial test)**
2. **Pseudo-exhaustive test**

Properties of exhaustive tests

1. **Advantages** (concerning the stuck at fault model):
   - test pattern generation is not needed
   - fault simulation is not needed
   - no need for a fault model
   - redundancy problem is eliminated
   - single and multiple stuck-at fault coverage is 100%
   - easily generated on-line by hardware
2. **Shortcomings:**
   - long test length (2^n patterns are needed, n - is the number of inputs)
   - CMOS stuck-open fault problem

LFSR

LFSR – Linear feedback shift register, hardware that generates pseudo-random pattern sequence



Characteristic Polynomial: $P(x) = 1 + x^3 + x^4$

Properties of Polynomials

- ***Irreducible polynomial*** **– cannot be factored, is divisible only by itself**
- ***Irreducible polynomial*** **of degree *n* is characterized by:**
  - **An odd number of terms including 1 term**
  - **Divisibility into $1 + x^k$, where $k = 2^n - 1$**
- **Any polynomial with all even exponents can be factored and hence is *reducible***
- **An *irreducible polynomial* is *primitive* if it divides the polynomial $1+x^k$ for $k = 2^n - 1$, but not for any smaller positive integer k**
- **Only *primitive* polynomials of an *n-bit* LFSR generates maximum possible *unique* patterns $2^n - 1$**

Reciprocal Polynomial

The reciprocal polynomial of *P(*X) is defined by:

1. $P^*(x) = x^n \cdot P(1/x)$
2. $X^a \Rightarrow X^{n-a}$

Example:

The reciprocal of polynomial $P_3(x) = 1 + x + x^3$ is

1. $P*(x) = x^3 \cdot (1 + \frac{1}{x} + \frac{1}{x^3}) = x^3 + x^2 + 1$
2. $P*(x) = x^{3-0} + x^{3-1} + x^{3-3} = x^3 + x^2 + 1$

The reciprocal of a primitive polynomial is also primitive

Primitive Polynomials

Number of primitive polynomials of degree *N*

N	No
1	1
2	1
4	2
8	16
16	2048
32	67108864

Table of primitive polynomials up to degree 31

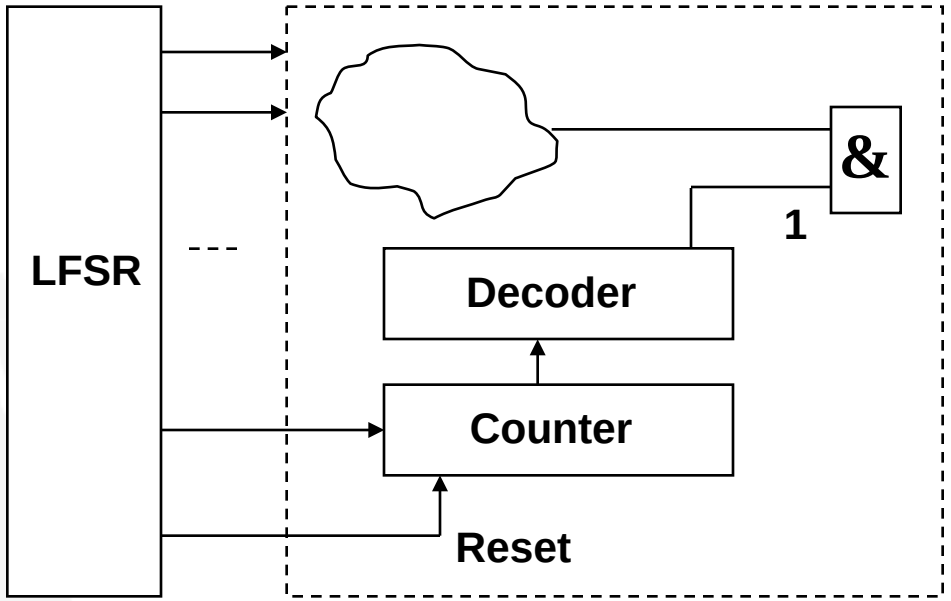
N	Primitive Polynomials
1,2,3,4,6,7,15,22	$1 + X + X^n$
5,11, 21, 29	$1 + X^2 + X^n$
10,17,20,25,28,31	$1 + X^3 + X^n$
9	$1 + X^4 + X^n$
23	$1 + X^5 + X^n$
18	$1 + X^7 + X^n$
8	$1 + X^2 + X^3 + X^4 + X^n$
12	$1 + X + X^3 + X^4 + X^n$
13	$1 + X + X^4 + X^6 + X^n$
14, 16	$1 + X + X^3 + X^4 + X^n$

Problems with Pseudorandom Test

The main motivations of using random patterns are:

- **low generation cost**
- **high initial efeciency**

Problem: **low fault coverage**



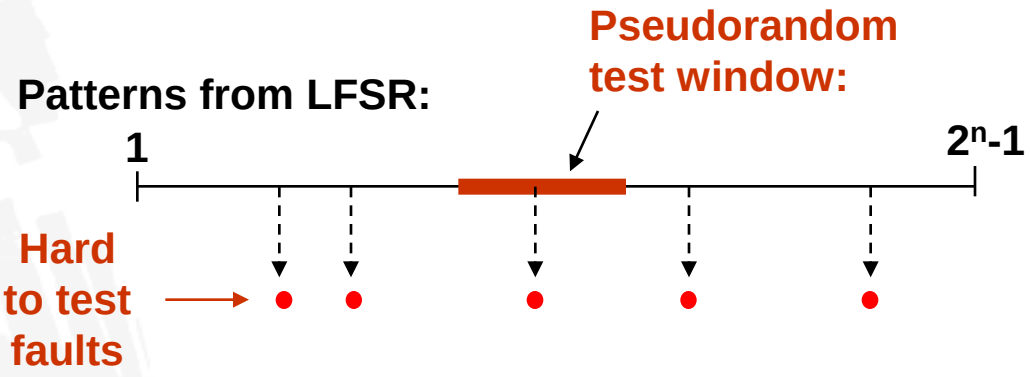
If Reset = 1 signal has probability 0,5 then counter will not work and 1 for AND gate may never be produced

Problems with BIST: Hard-To-Test-Faults

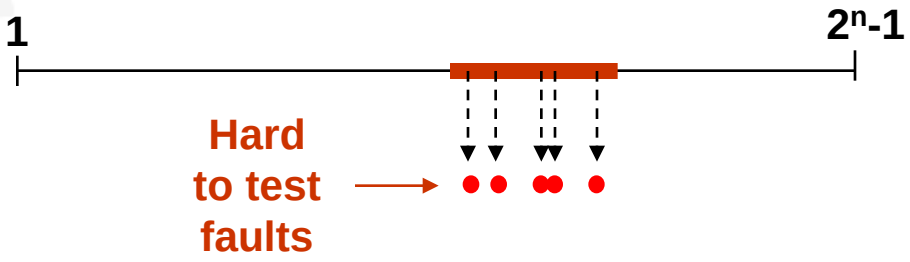
The main motivations of using random patterns are:

- low test generation cost
- high initial efficiency

Problem: Low fault coverage

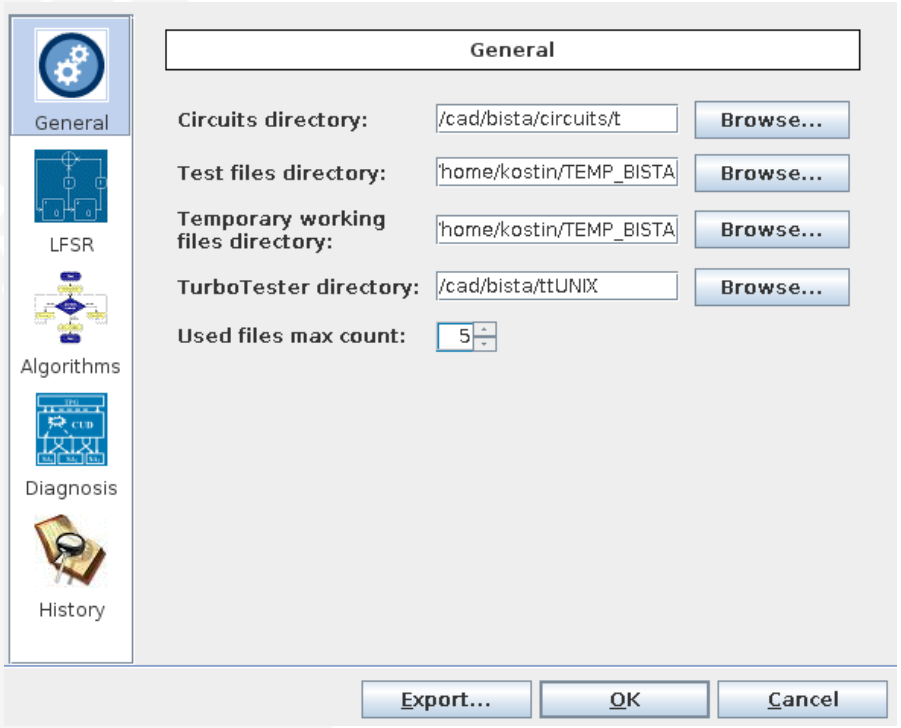


Dream solution: Find LFSR such that:



NB! For using BISTA in Linux classroom

Check for BISTA settings

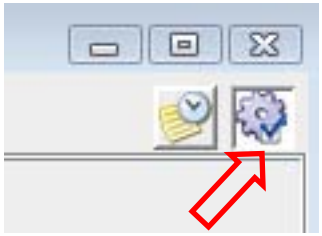


Circuits directory: **/cad/bista/circuits**
Test files directory: **/t0...../TEMP_BISTA (t0... – your home directory)**
Temporary directory: **/t0...../TEMP_BISTA**
Turbotester directory: **/cad/bista/ttUNIX**

PRPG Panel Tips

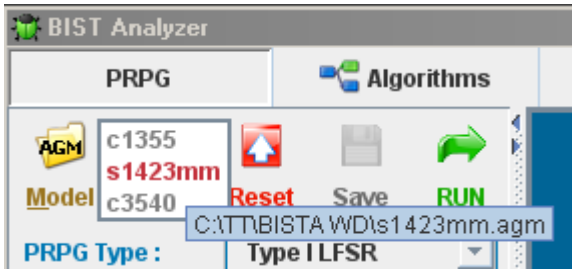
Settings panel:

Used to define program startup parameters



Selection tracing

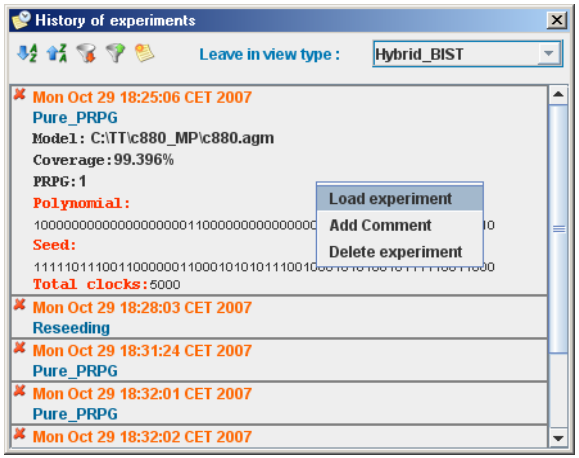
The model files chosen by user are traced and later could be reselected without searching their location in the file system



right click on model button

History panel:

reflects experiments and result performed during the session or restored from memory



left click – shows full info;
right click – load data

Loop detection

For identifying primitive polynomials

Algorithm Panel Tips (1)

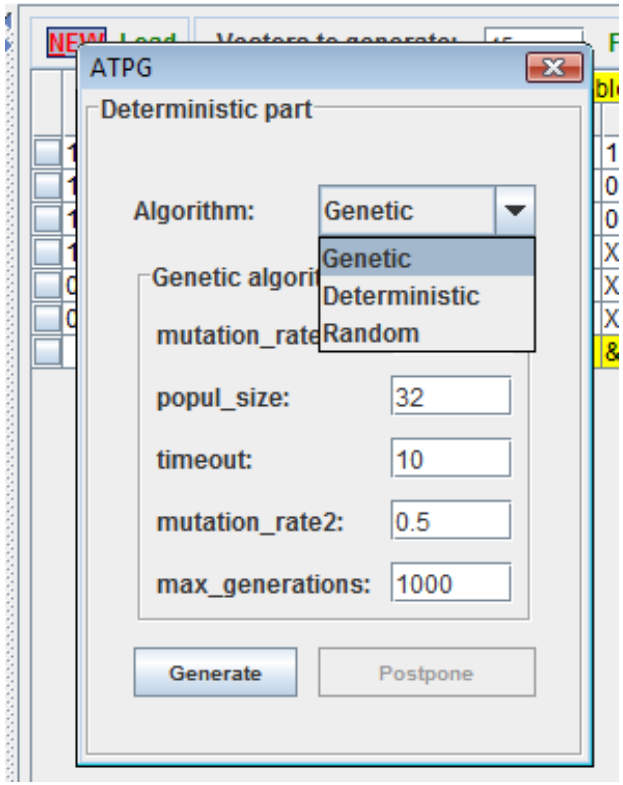
➢ Algorithms → New = ATPG

ATPG (Automated Test Pattern Generator)

- **Genetic**
- **Deterministic**
- **Random**

ATPG algorithms description:

- http://www.pld.ttu.ee/testing/labs/genetic.html
- http://www.pld.ttu.ee/testing/labs/generate.html
- http://www.pld.ttu.ee/testing/labs/random.html



Algorithm Panel Tips (2)

Looking for “good” seed (press “New or “Load”)

NEW Load | Vectors to generate: 45 | Fast | Find HTTF | Cover HTTF

	Vectors	% (Total)	%	Fault Table														
✔	11001	36,667	36,667	1	1	X	X	1	X	1	0	0	0	0	1	X	0	1
	11101	73,333	36,667	0	0	0	0	0	X	X	X	1	X	1	0	0	1	0
	11110	80,000	40,000	0	X	1	0	0	1	X	X	1	0	0	0	0	1	1
	10110	93,333	20,000	X	X	X	1	X	0	0	X	X	1	X	X	X	0	0
	00000	96,667	26,667	X	X	X	X	X	X	1	1	X	0	0	0	0	1	1
	00010	100,000	20,000	X	X	X	X	X	0	0	X	X	1	X	X	1	0	0
				&	&	&	&	&	&	&	&	&	&	&	&	&	&	&

- **Number of patterns to be considered: Fast–Medium–Thorough**
- **Find HTTF – finds hard-to-test faults**
- **Cover HTTF – shows pattens testing HTTF**
- **Selected vectors can be saved by pressing “Save selection”**
- **Any .tst file can be considered for looking HTTF (press “Load”)**