1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Testimise projekteerimine:

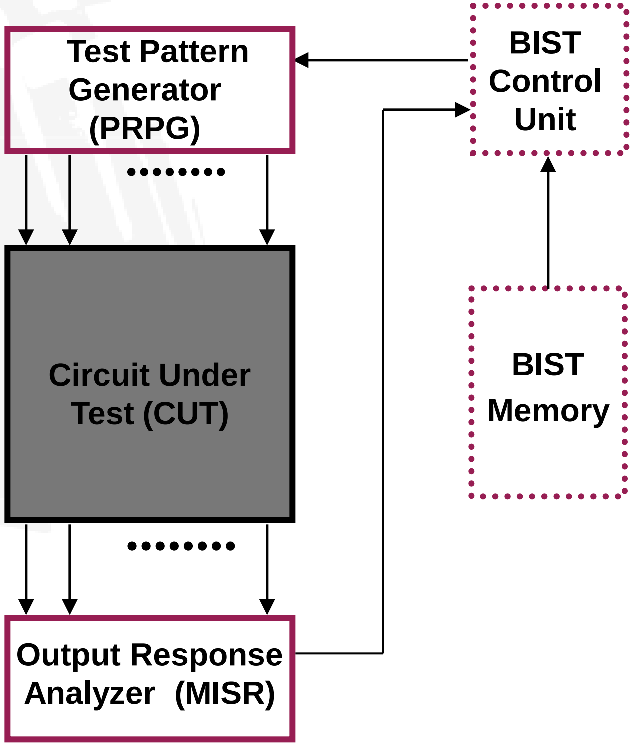
Labor 2
BIST Optimization

Sergei Kostin

BIST (Built-in Self Test)

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

Typical BIST Architecture



Built-In Self-Test

- **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

BIST Benefits

- **Faults tested:**
  - Single stuck-at faults
  - Delay faults
  - Single stuck-at faults in BIST hardware
- **BIST benefits**
  - Reduced testing and maintenance cost
  - Lower test generation cost
  - Reduced storage / maintenance of test patterns
  - Simpler and less expensive ATE
  - Can test many units in parallel
  - Shorter test application times
  - Can test at functional system speed

Economics – BIST Costs

- **Chip area overhead for:**
  - Test controller
  - Hardware pattern generator
  - Hardware response compacter
  - Testing of BIST hardware
- **Pin overhead -- At least 1 pin needed to activate BIST operation**
- **Performance overhead – extra path delays due to BIST**
- ***Yield loss* – due to increased chip area or more chips In system because of BIST**
- **Reliability reduction – due to increased area**
- **Increased BIST hardware complexity – happens when BIST hardware is made testable**

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

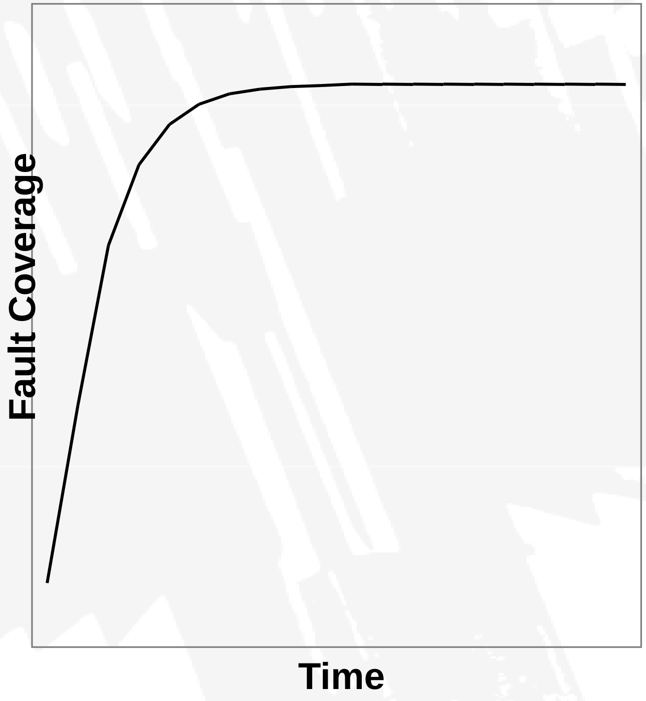
2. Shortcomings:

- long test length (2^n patterns are needed, n - is the number of inputs)
- CMOS stuck-open fault problem

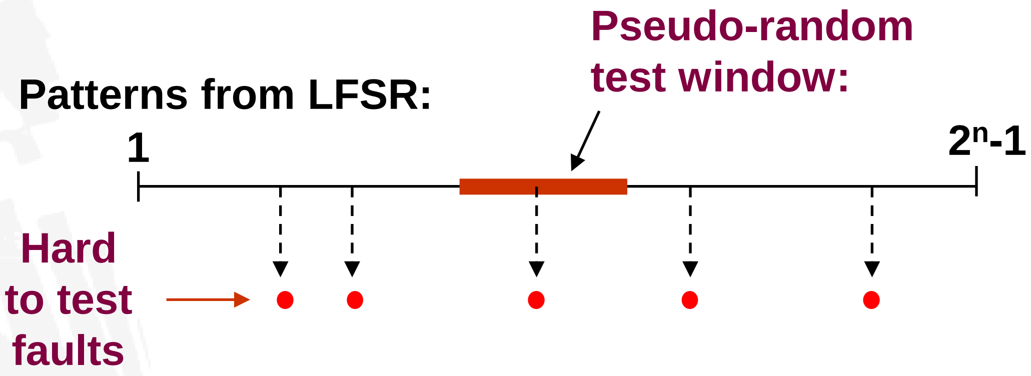
Problems with BIST: Hard to Test Faults

The main motivations of using random patterns are:

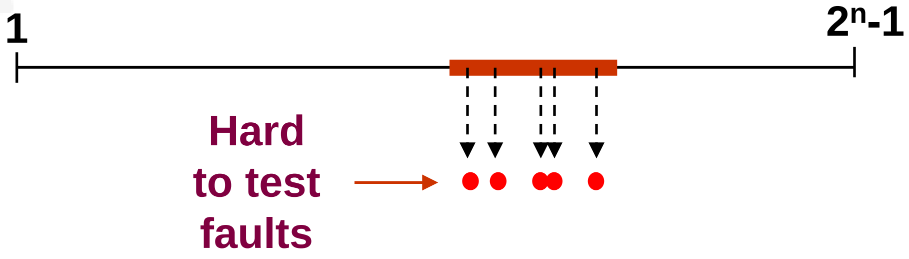
- low test generation cost
- high initial efficiency



Problem: Low fault coverage



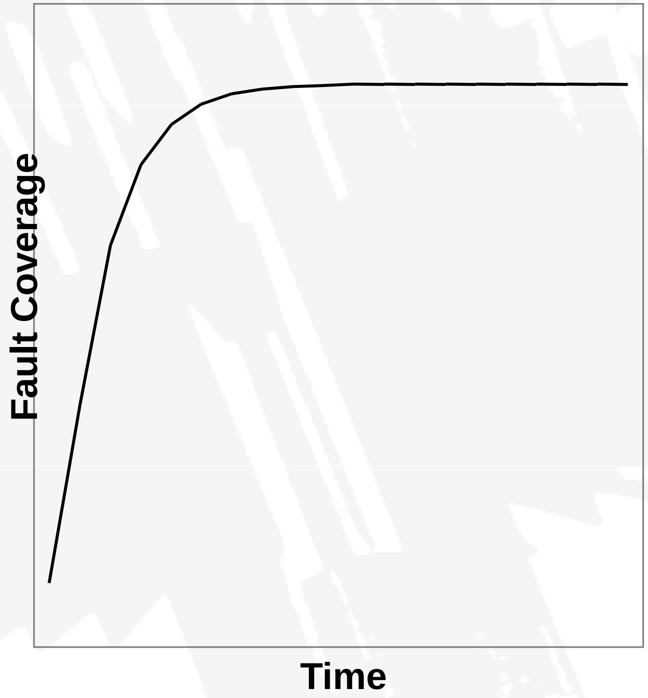
Dream solution: Find LFSR such that:



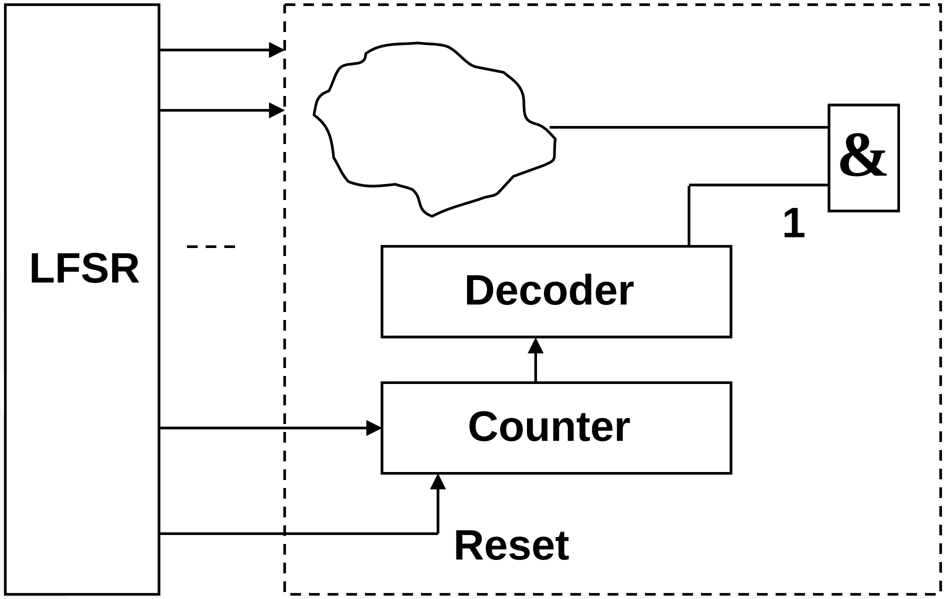
Problems with Pseudo-Random Test

The main motivations of using random patterns are:

- low generation cost
- high initial efeciency

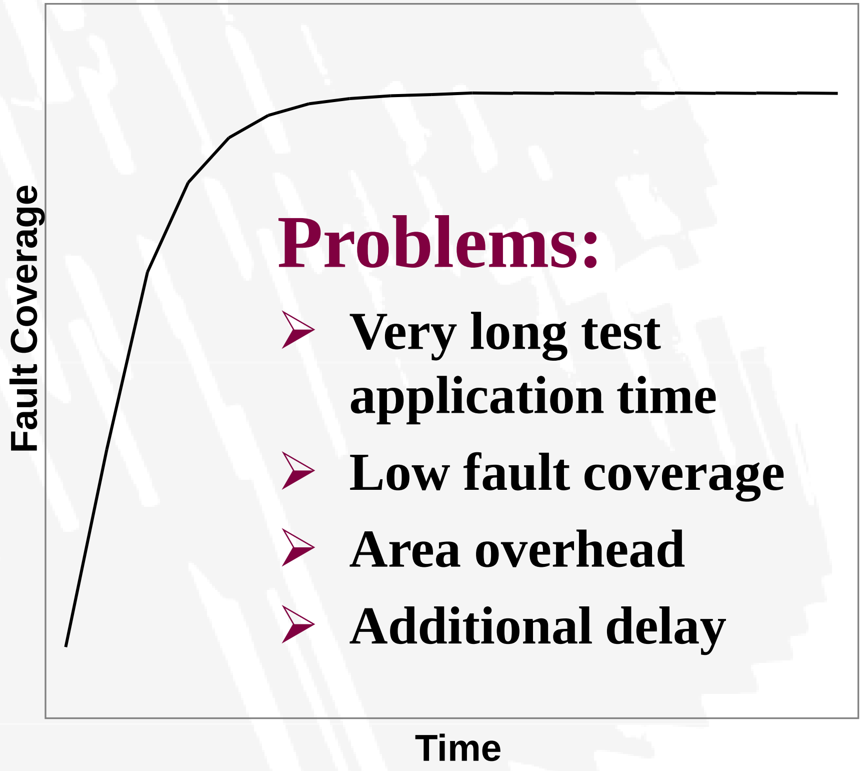


Problem: low fault coverage



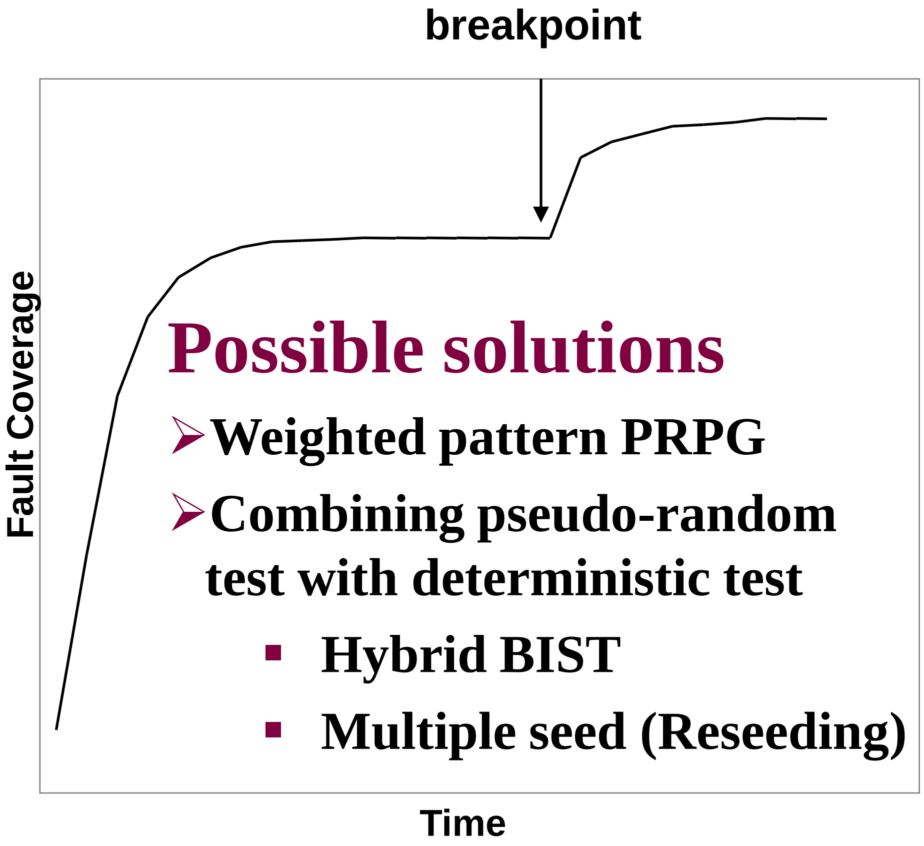
If Reset = 1 signal has probability 0,5 then counter will not work and
1 for AND gate may never be produced

Pseudo-Random Test Generation by LFSR



Problems:

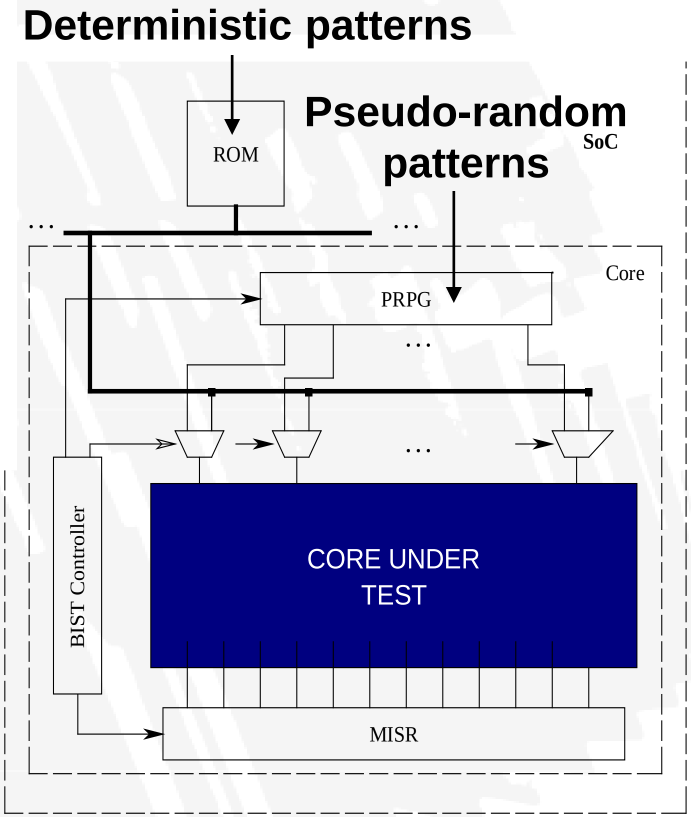
- Very long test application time
- Low fault coverage
- Area overhead
- Additional delay



Possible solutions

- Weighted pattern PRPG
- Combining pseudo-random test with deterministic test
  - Hybrid BIST
  - Multiple seed (Reseeding)

Hybrid Built-In Self-Test



Hybrid test set contains pseudo-random and deterministic vectors

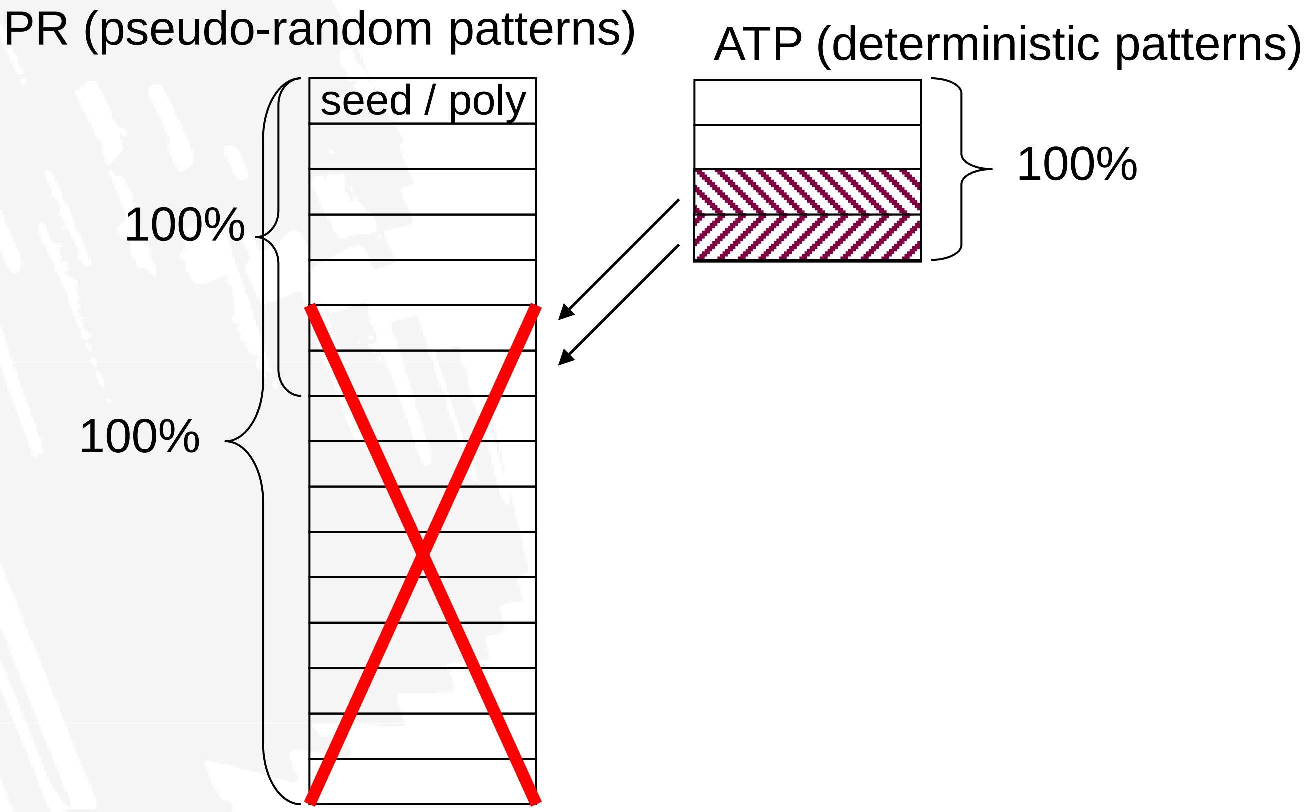
Pseudo-random test is improved by a stored test set which is specially generated to target the random resistant faults

Optimization problem:

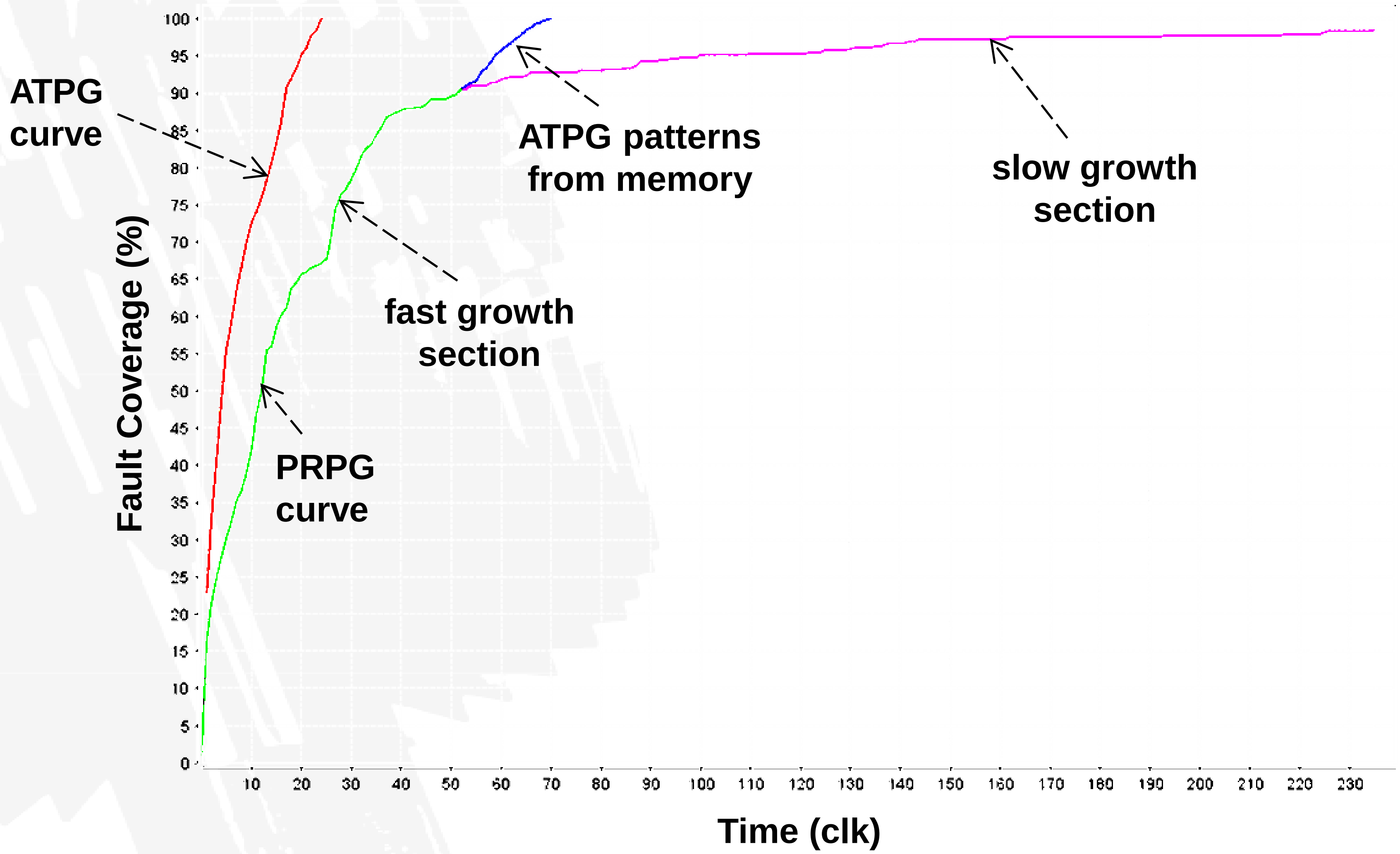
Where should be this breakpoint?

Pseudo-random Test	Determ. Test

Hybrid BIST Technique



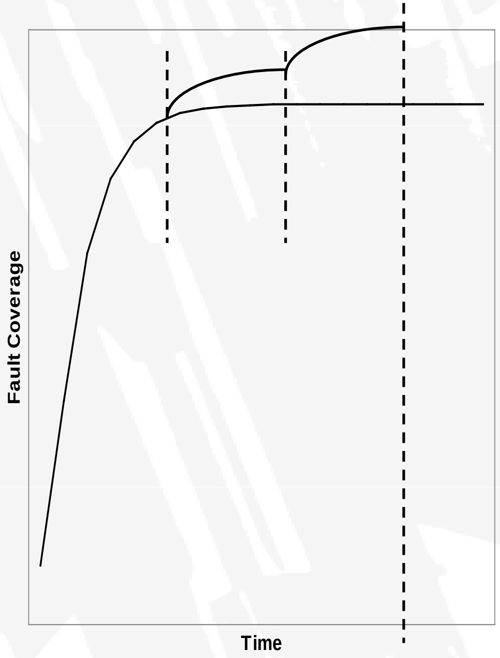
BIST Optimization Challenges



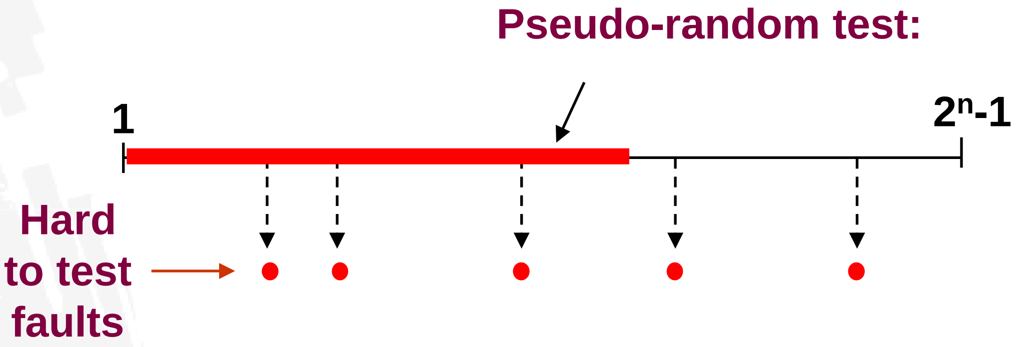
Reseeding (Multiple Seeds)

The main motivations of using random patterns are:

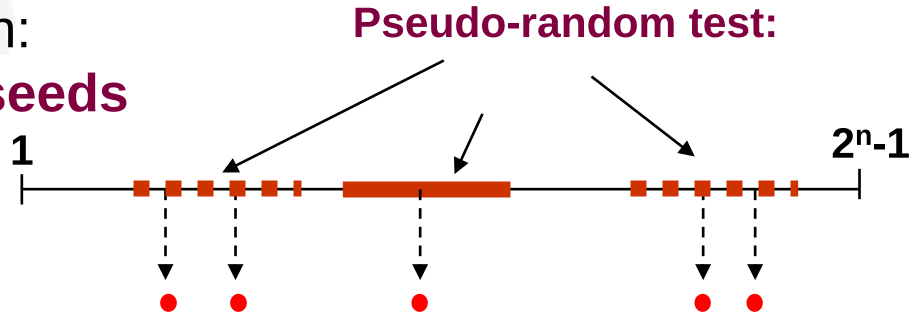
- low generation cost
- high initial efeciency



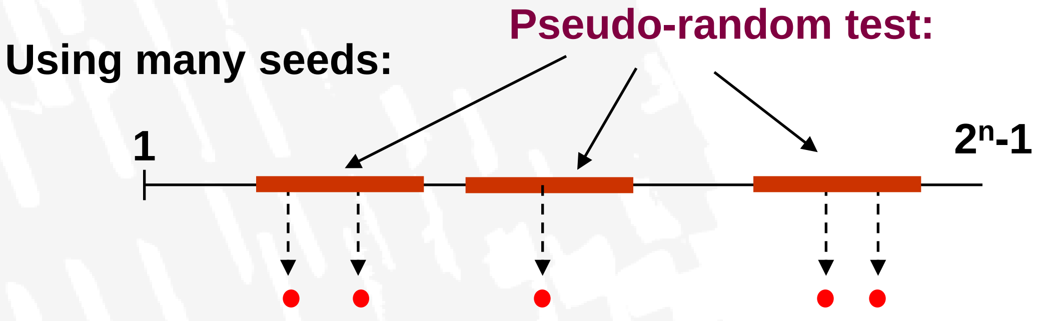
Problem: low fault coverage → long PR test



Solution: many seeds



Reseeding Optimization Problem

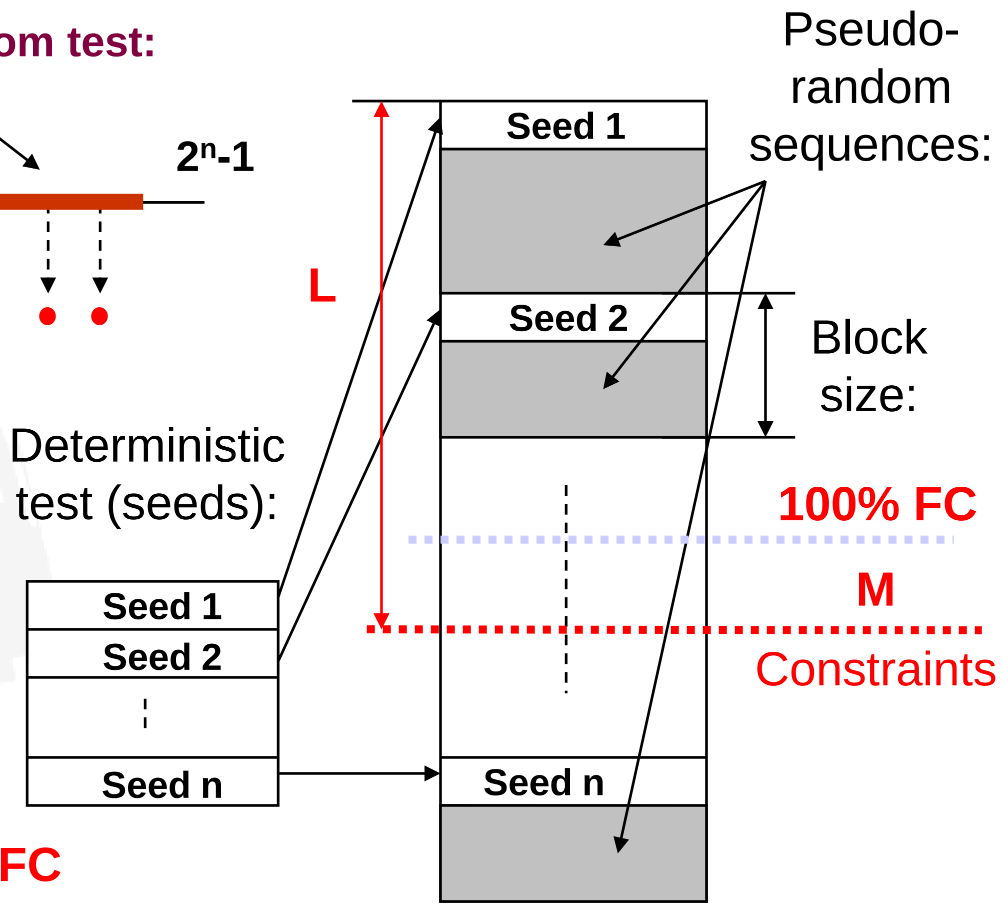


Problems:

How to calculate the number and size of blocks?

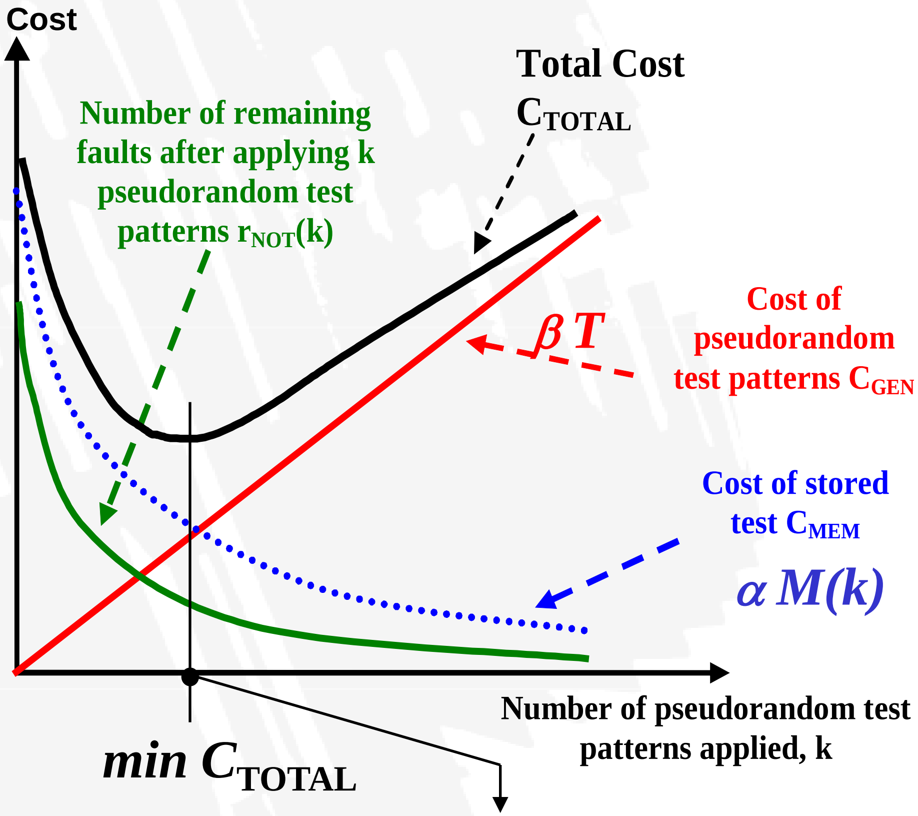
Which deterministic patterns should be the seeds for the blocks?

Minimize L at given M and 100% FC



Cost Calculation for Hybrid BIST

$$C_{TOTAL} = \beta T + \alpha M(k)$$



Pseudorandom Test | **Det. Test**

PR test length

# faults not detected

# tests needed

k	$r_{DET}(k)$	$r_{NOT}(k$	$FC(k)$	$t(k)$
1	155	839	15.6%	104
2	76	763	23.2%	104
3	65	698	29.8%	100
4	90	608	38.8%	101
5	44	564	43.3%	99
10	104	421	57.6%	95
20	44	311	68.7%	87
50	51	218	78.1%	74
100	16	145	85.4%	52
200	18	114	88.5%	41
411	31	70	93.0%	26
954	18	28	97.2%	12
1560	8	16	98.4%	7
2153	11	5	99.5%	3
3449	2	3	99.7%	2
4519	2	1	99.9%	1
4520	1	0	100.%	0

Variant 5

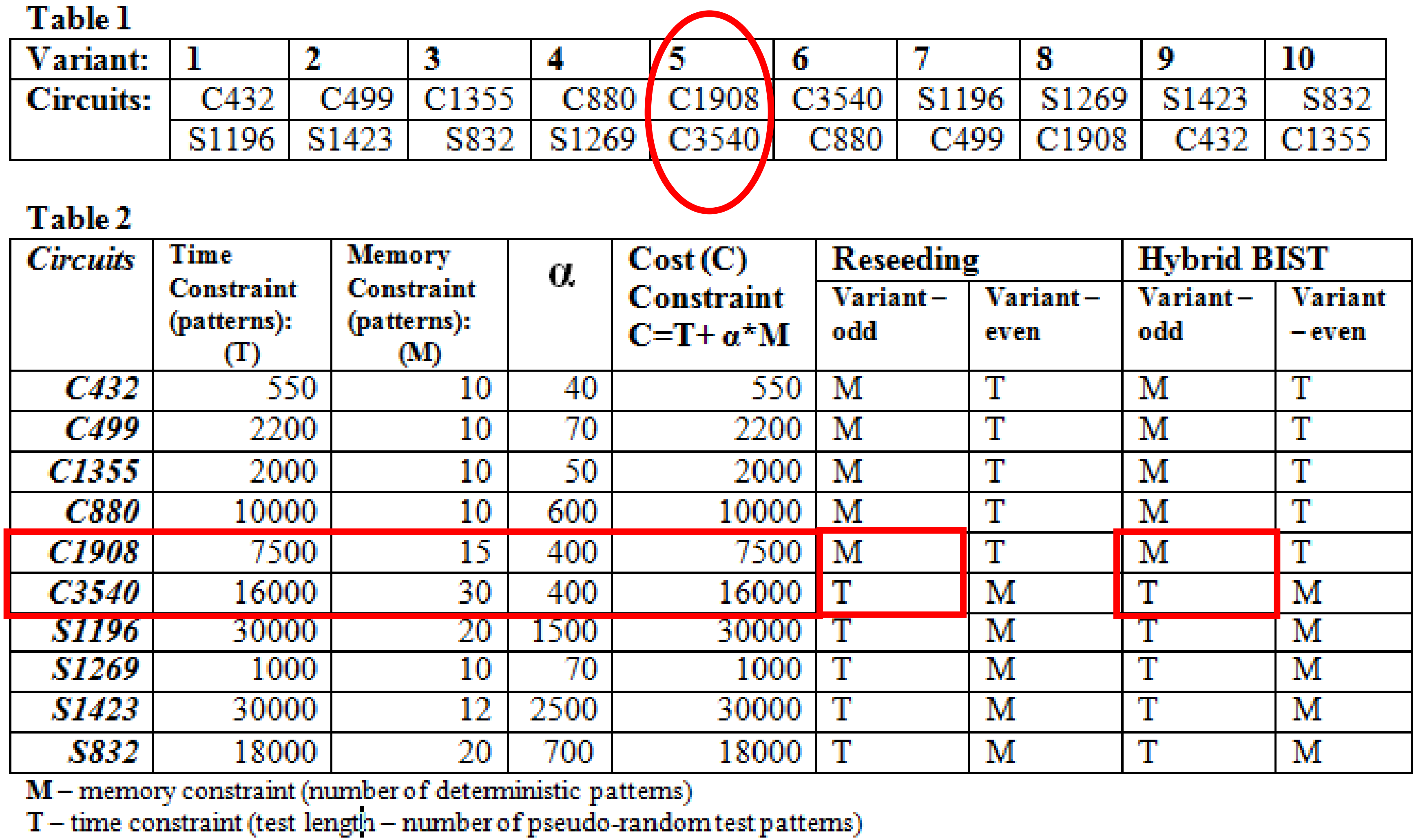


Table 1

Variant:	1	2	3	4	5	6	7	8	9	10
Circuits:	C432	C499	C1355	C880	C1908	C3540	S1196	S1269	S1423	S832
	S1196	S1423	S832	S1269	C3540	C880	C499	C1908	C432	C1355

Table 2

Circuits	Time Constraint (patterns): (T)	Memory Constraint (patterns): (M)	α	Cost (C) Constraint C=T+ α*M	Reseeding		Hybrid BIST	
					Variant – odd	Variant – even	Variant – odd	Variant – even
C432	550	10	40	550	M	T	M	T
C499	2200	10	70	2200	M	T	M	T
C1355	2000	10	50	2000	M	T	M	T
C880	10000	10	600	10000	M	T	M	T
C1908	7500	15	400	7500	M	T	M	T
C3540	16000	30	400	16000	T	M	T	M
S1196	30000	20	1500	30000	T	M	T	M
S1269	1000	10	70	1000	T	M	T	M
S1423	30000	12	2500	30000	T	M	T	M
S832	18000	20	700	18000	T	M	T	M

M – memory constraint (number of deterministic patterns)

T – time constraint (test length – number of pseudo-random test patterns)

ISCAS’ 85 benchmark circuit

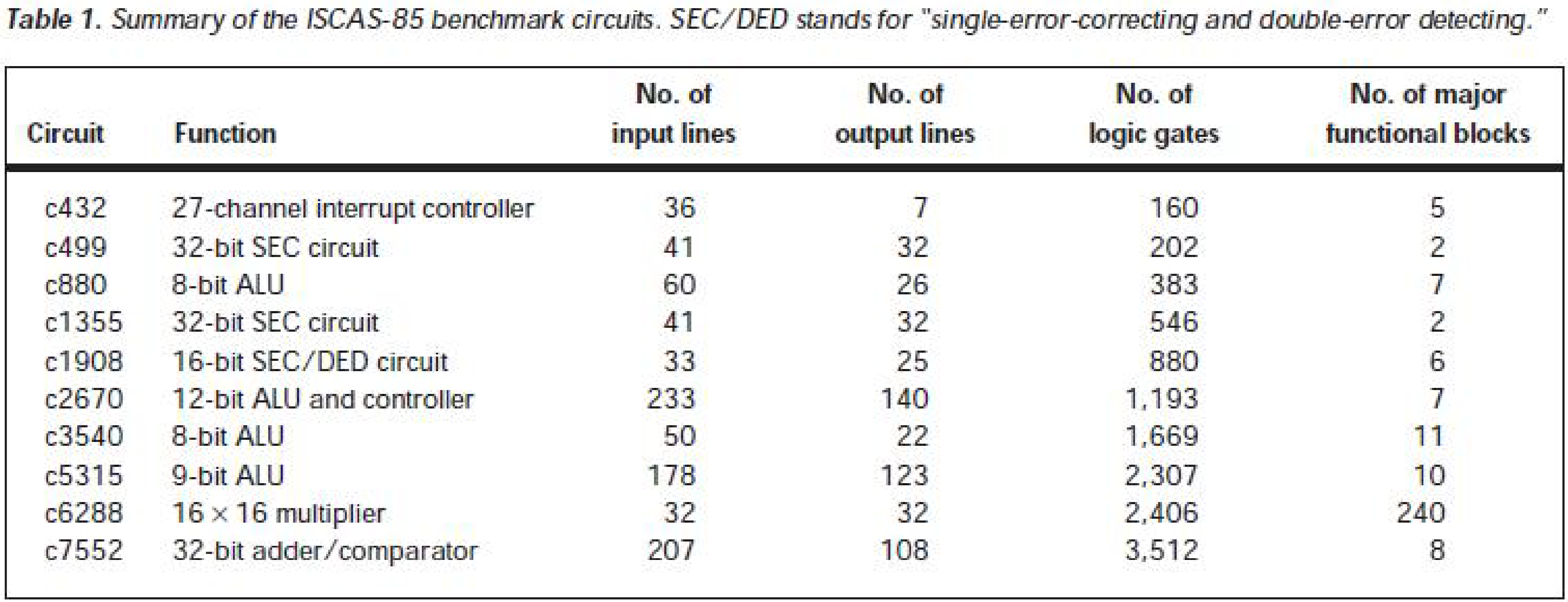


Table 1. *Summary of the ISCAS-85 benchmark circuits. SEC/DED stands for "single-error-correcting and double-error detecting."*

Circuit	Function	No. of input lines	No. of output lines	No. of logic gates	No. of major functional blocks
c432	27-channel interrupt controller	36	7	160	5
c499	32-bit SEC circuit	41	32	202	2
c880	8-bit ALU	60	26	383	7
c1355	32-bit SEC circuit	41	32	546	2
c1908	16-bit SEC/DED circuit	33	25	880	6
c2670	12-bit ALU and controller	233	140	1,193	7
c3540	8-bit ALU	50	22	1,669	11
c5315	9-bit ALU	178	123	2,307	10
c6288	16 × 16 multiplier	32	32	2,406	240
c7552	32-bit adder/comparator	207	108	3,512	8

Task 1: Pseudo-random test generation

- Find out the maximum fault coverage for given circuits applying tuned ATPG.

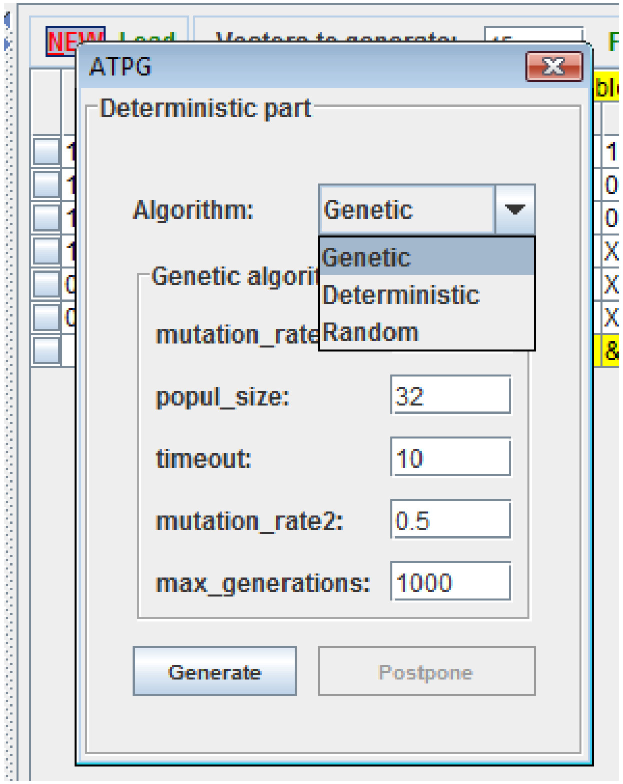
➢ Algorithms → New = ATPG

ATPG (Automated Test Pattern Generator)

- Genetic
- Deterministic
- Random

ATPG algorithms description:

- Turbo Tester v02.10.pdf (edu.pld.ttu.ee → minu materjalid)
  - 4.2 Test Pattern Generation


Algorithm: Genetic

popul_size: 32

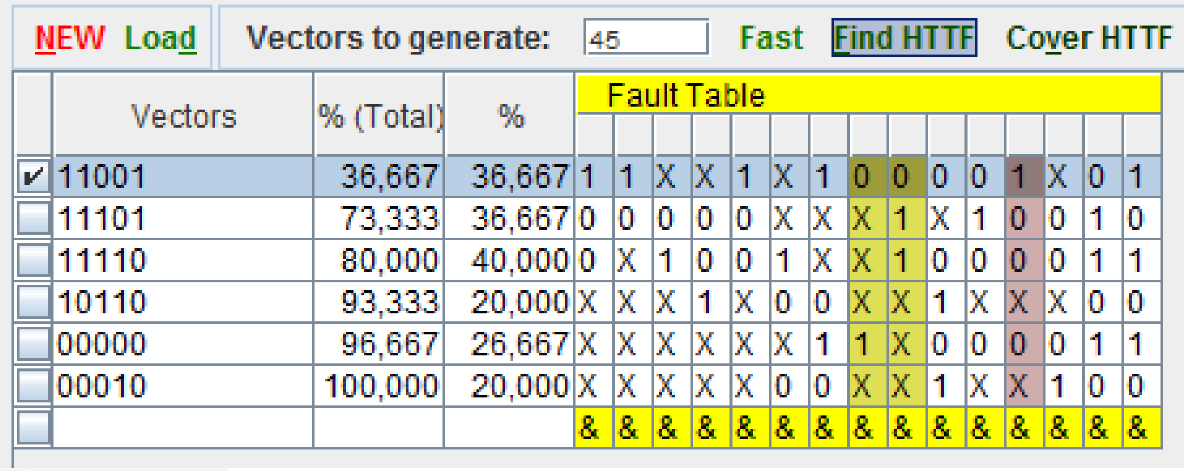
timeout: 10

mutation_rate2: 0.5

max_generations: 1000

Task 1: Pseudo-random test generation

- **Choose "good" seed and polynomial to generate effective pseudo-random test.**
  - "good" polynomial: random or primitive
  - "good" seed: pattern testing HTTF (hard-to-test faults) (select from ATPG test)



Vectors to generate: 45

	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
☐				&	&	&	&	&	&	&	&	&	&	&	&	&	&	&

Task 1: Pseudo-random test generation

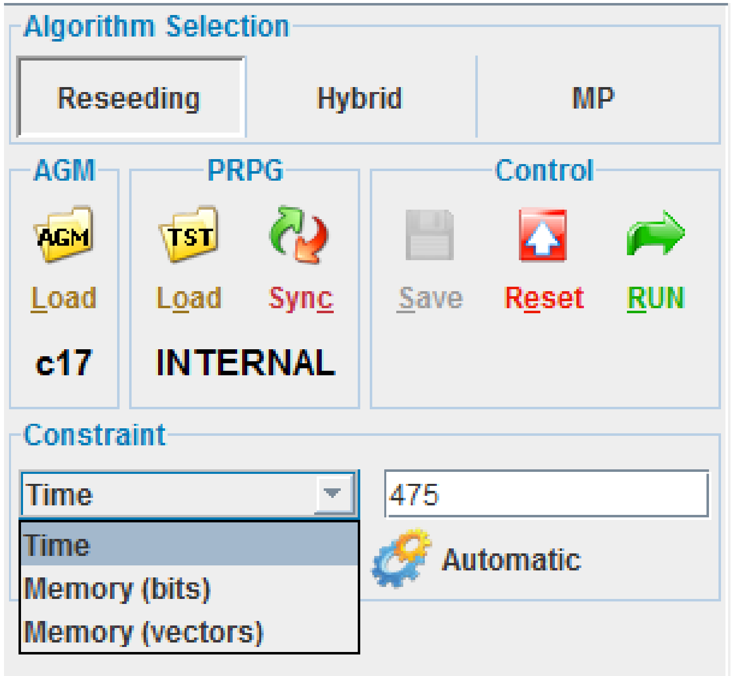
- **Pseudo-random test sequence must have**
  - maximum fault coverage (same as for ATPG)
  - length of test must be in the range **T<= length<=1,2T** where **T – Time Constraint** from Table 2.
- **Example:** c1908 → T = 7500 and 1,2T = 1,2*7500 = 9000 → 7500 <= test length <= 9000
- **Hint:** When generating a test put the number of clock cycles equal to 1,2T
  - if maximum fault coverage achieved within given test then cut out last patterns that do not give additional fault coverage
- **Use Type I and Type II generators.**
- **Fill in the table**

Task 2 and 3

- **Reseeding algorithm:**
  - **Find complete test (max. fault coverage) for given circuit, using the best PRPG (Type I LFSR) test sequence achieved in task 1.**
  - **Perform experiments stepping through the constraint values and choose 5 results (test must comply with the constraints specified in your variant). Example: c1908→ M = 1, 2, 3…10 (min.step = 1); T = 7500, 7000, 6500… (min.step = at least 5% of T).**

- **Hybrid BIST algorithm:**
  - **Find complete test (max. fault coverage) for given circuit, using the best PRPG (Type I LFSR) test sequence achieved in task 1.**
  - **Perform experiments and choose 5 evenly distributed results (test must comply with the constraints specified in your variant).**

Reseeding and Hybrid Algorithms



AGM: c17

PRPG: INTERNAL

Constraint: Time — 475

- Time
- Memory (bits)
- Memory (vectors)

➢ PRPG

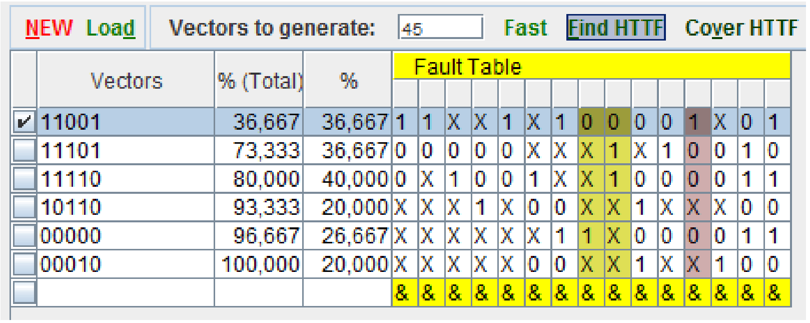
- Load → use saved sequence of pseudo-random test patterns
- Sync → use generated in PRPG panel sequence of test patterns

➢ Constraint

- Time → test length (number of pseudo-random test patterns)
- Memory (vectors) → number of deterministic test patterns stored in memory

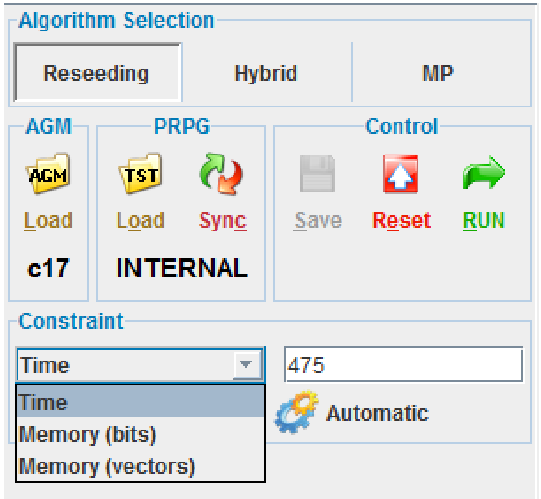
Reseeding and Hybrid Algorithms

- ➢ **Load model (AGM panel)**
- ➢ **Load or Sync PR test (PRPG panel)**
- ➢ **Generate ATPG test (press New or Load button)**
- ➢ **Select Time or Memory(vectors) Constraint and enter the constraint**
- ➢ **Run**



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
☐				&	&	&	&	&	&	&	&	&	&	&	&	&	&	&



Algorithm Selection

Reseeding | Hybrid | MP

AGM: Load — c17

PRPG: Load, Sync — INTERNAL

Control: Save, Reset, RUN

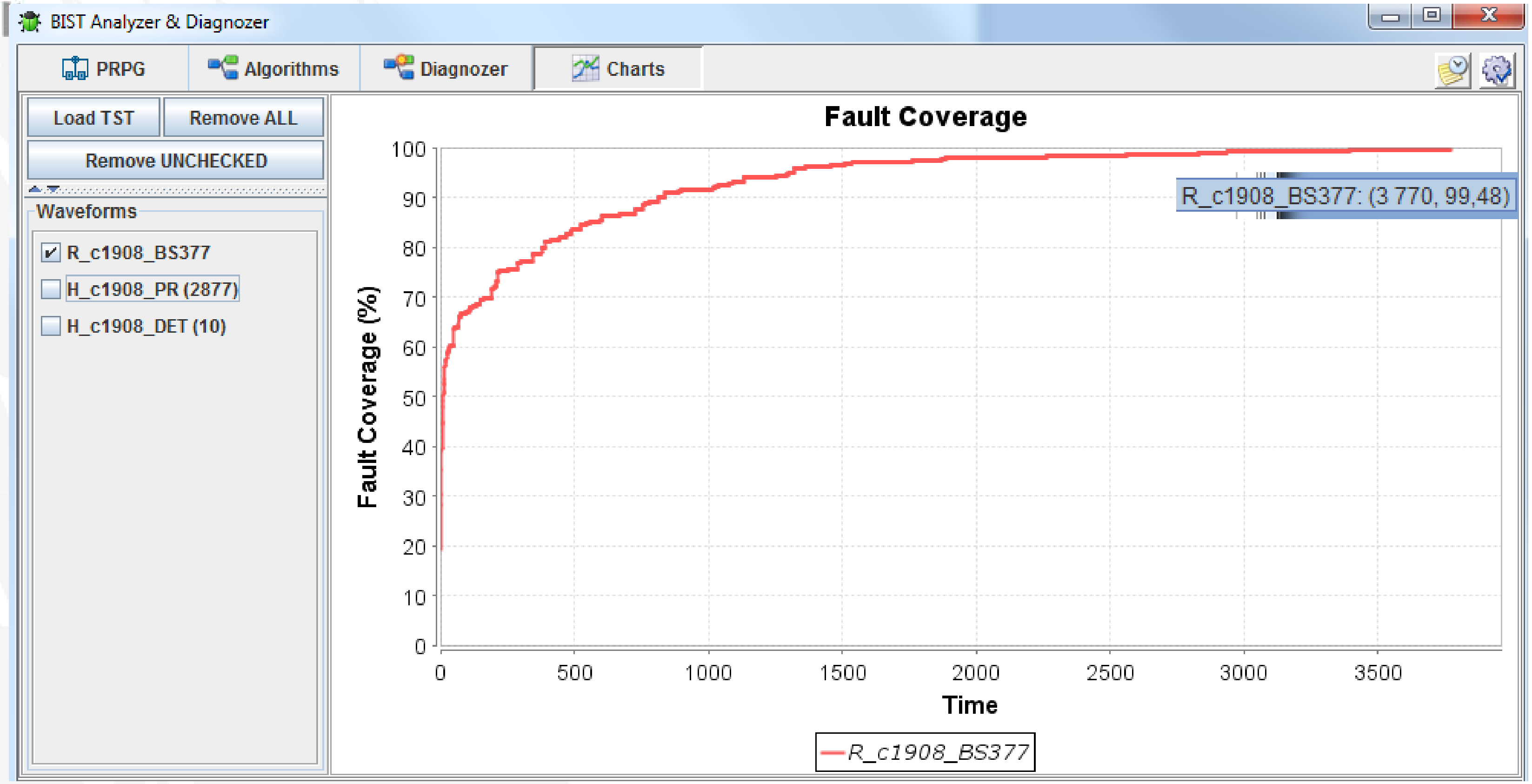
Constraint: Time — 475

Time
Memory (bits)
Memory (vectors)

Automatic

In order to Run Reseeding or Hybrid Algorithm with another constraint: just change constraint and press **Run**

Reseeding: calculating BIST cost

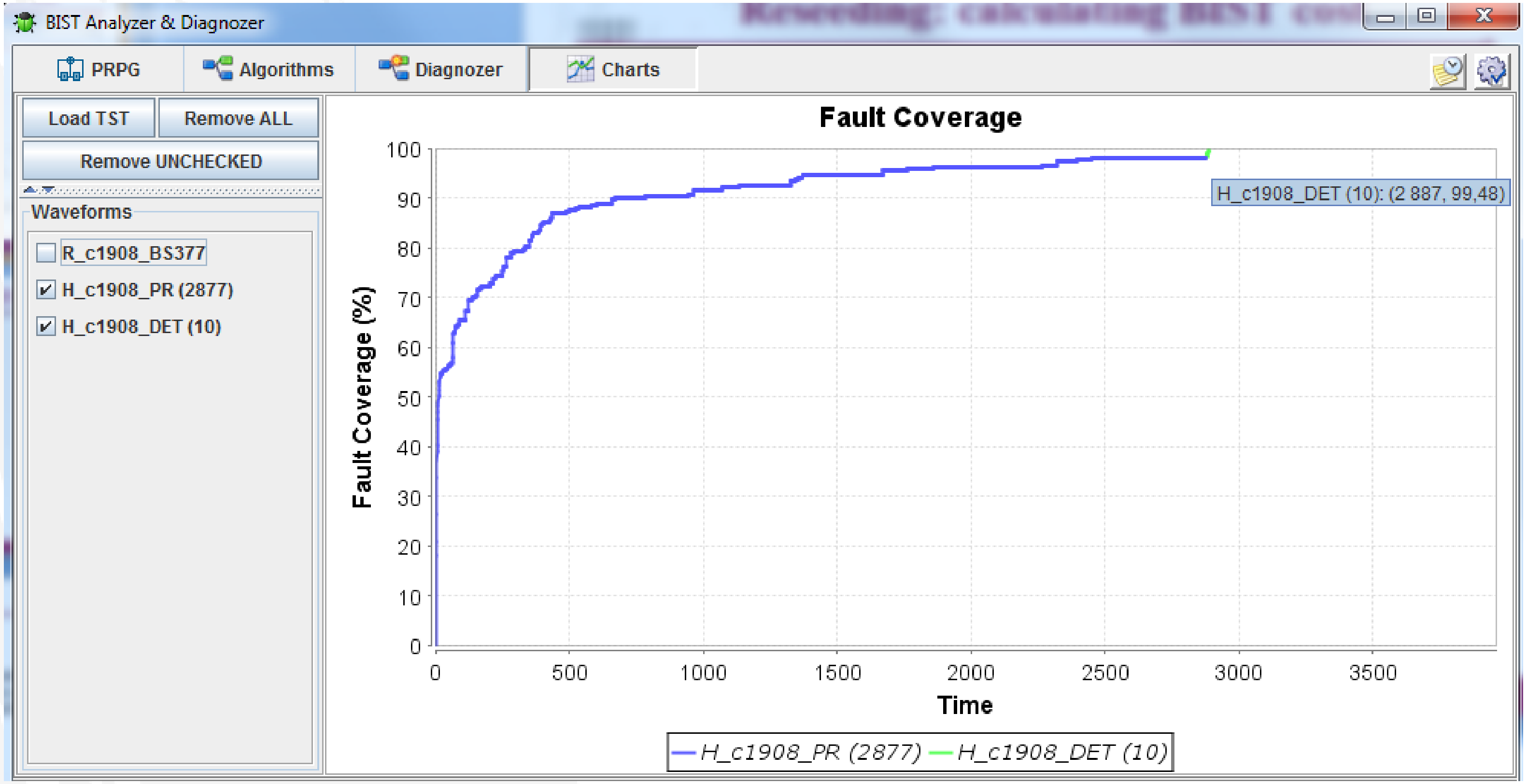


Test Length (T) = 3770, **FV** = 99,48%

Block size = 377 → **M** = 3770/377 – 1 = 9

Cost = 3770 + 400*9 = 7370

Hybrid: calculating BIST cost



M = 10, **FV** = 99,48%

Test Length (T) = 2887 + 10 = 2897,

Cost = 2897 + 400*10 = 6897

Fill in the Result Table

Circuit 1: C1908

Reseeding				Hybrid BIST			
Calculated Cost:	**Test Length**	**Memory Vectors**	**Fault Coverage**	**Calculated Cost**	**Test Length**	**Memory Vectors**	**Fault Coverage**
7370	3770	9	99,48	6897	2897	10	99,48
7850	4250	9	99,48	6100	4500	4	99,48

Memory Constraint: min. step is 1

- Example: c1908→ constraint = 15 → M = 1,2,3,...,15

Time Constraint: min. step is at least 5% of constraint and number of memory vectors must change

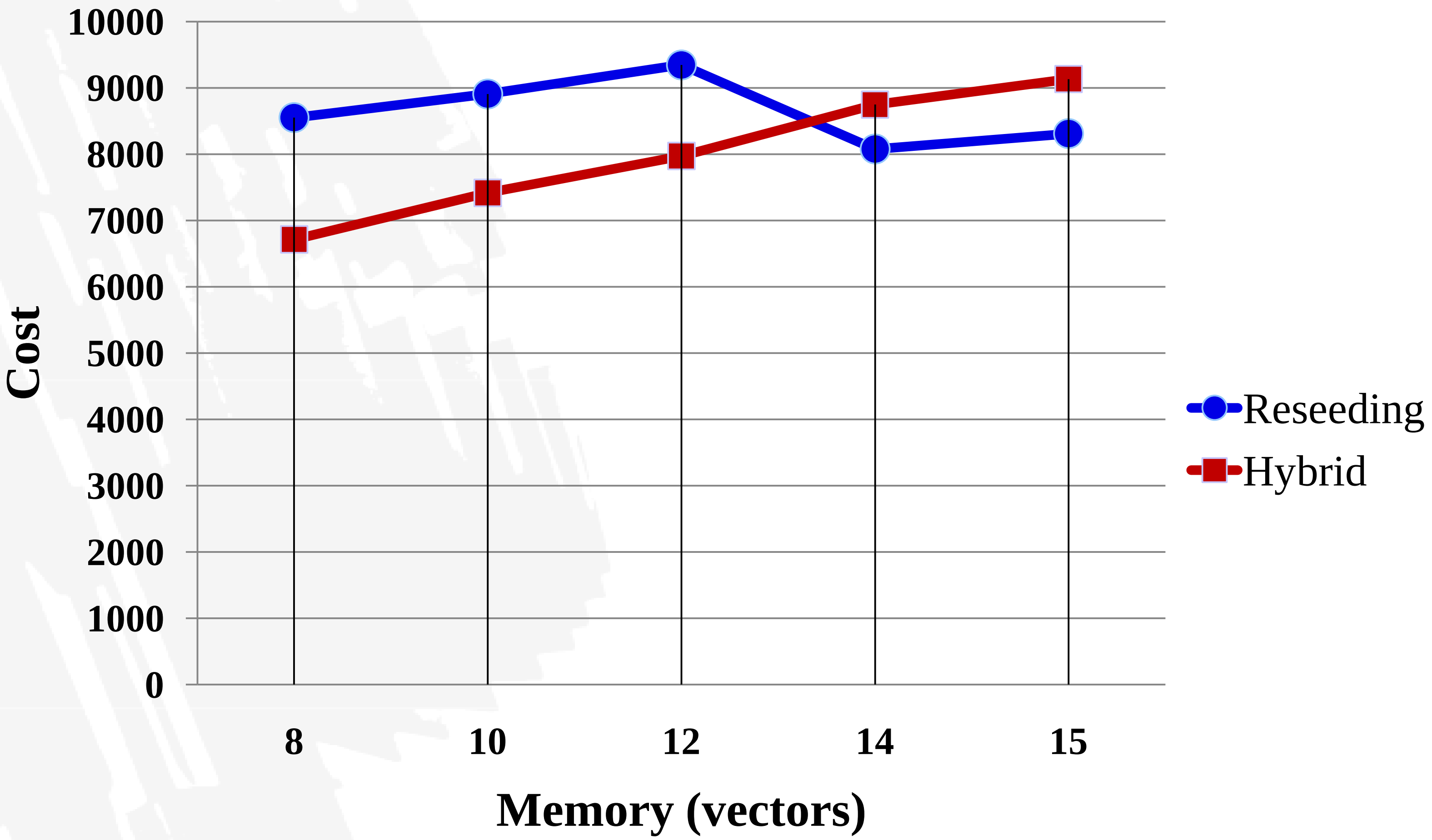
- Example: c1908→ constraint = 7500 → T = 7500, 7000, 6500...
- **NB!** if cannot find 5 satisfactory tests for Reseeding, decrease min. step or use Memory constraint instead

NB! Try to use the same constraints for both algorithms → this helps to compare effectiveness of Reseeding and Hybrid algorithms

Task 4

- **Results evaluation:**
  - **According to the marginal results obtained in task 2 and 3 construct the Cost curves on the same graph for Reseeding and Hybrid BIST algorithms.**
  - **Compare results of Reseeding and Hybrid algorithms.**

BIST Cost Curves for Circuit c1908


Cost
Memory (vectors)
Reseeding
Hybrid