Transition Test Volume Reduction Using Test Point Insertion at RTL

Kedarnath J. Balakrishnan

Advanced Micro Devices, Austin

Outline

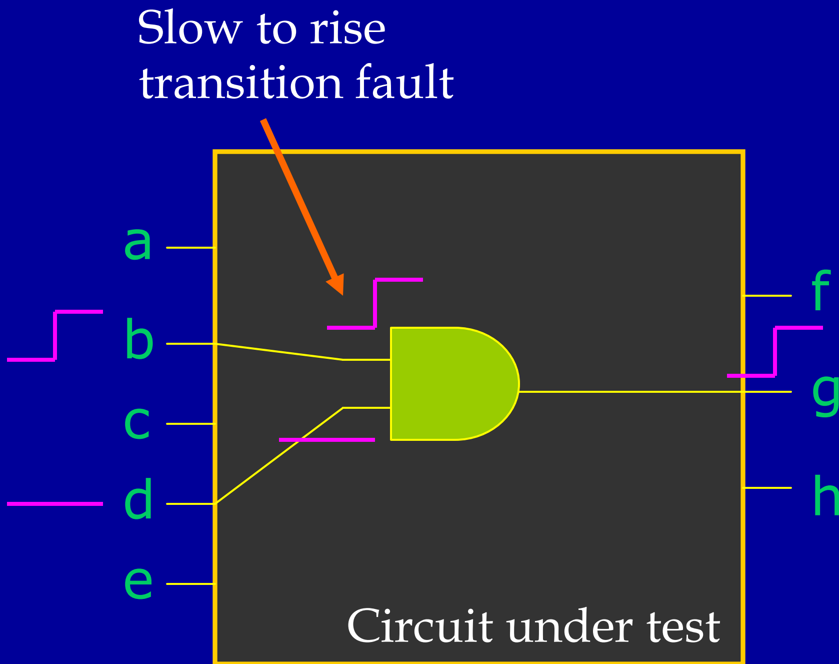
- **Motivation**
- **Background**
  - **Transition Fault Testing**
  - **SATisfiability**
- **Proposed Test Points**
- **Identifying Test Points at RTL**
  - **Using SATisfiability**
- **Experimental Setup and Results**
- **Conclusions and Future Work**

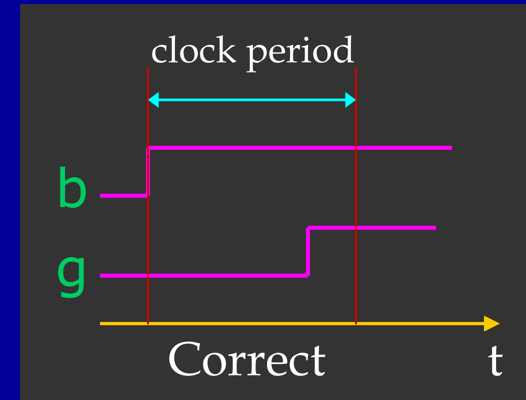
Motivation

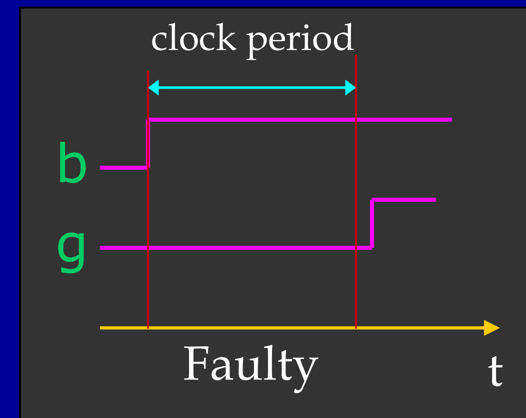
- **Delay fault testing part of regular DFT flow**
  - **a requirement for DSM designs**
- **Transition pattern data volume 4x-6x more than stuck-at**
- **Methods to reduce test data volume**
  - **Test Compression**
- **What can be done in the design?**
  - **Test Point Insertion**

Background: Transition Testing

- **Two pattern test to detect transition delay at gate**





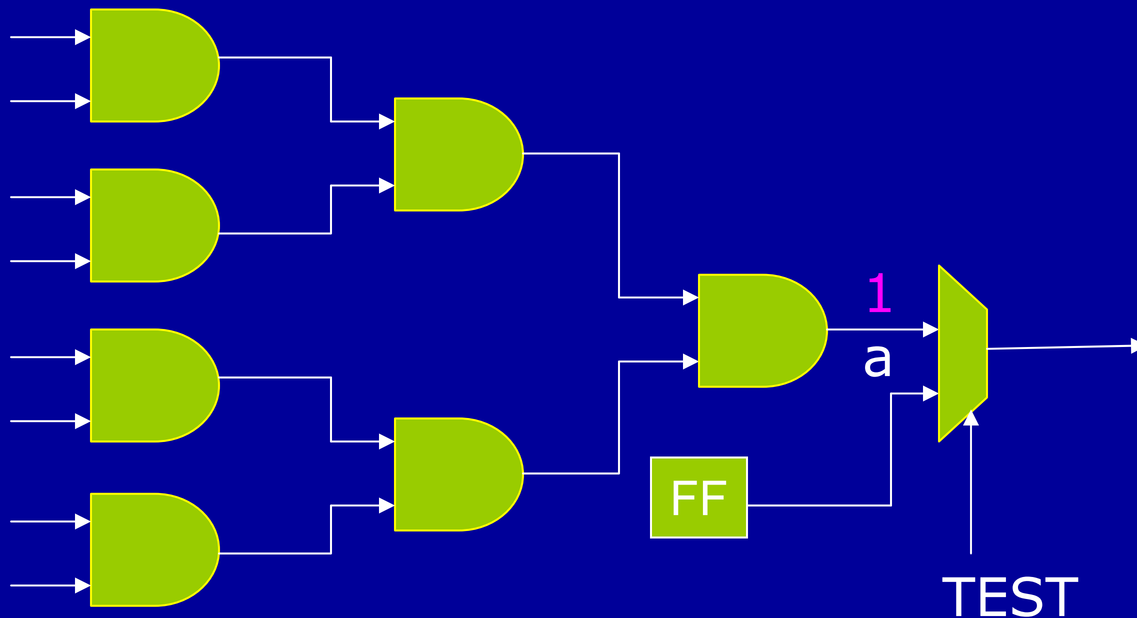


Transition Fault Testing

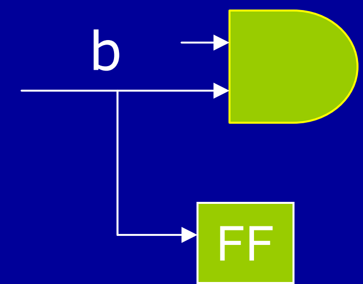
- **Launch-on-shift (*skewed load*)**
  - **Second pattern generated by shifting first pattern**
  - **Requires scan_en to be shifted at speed**
  - **Higher fault coverage**
- **Launch-on-capture (*broad-side*)**
  - **Second pattern is circuit response of first pattern**
  - **Easier to implement**
- **Enhanced Scan**
  - **Two flip-flops to store both patterns**
  - **Hardware overhead**

Background: Test Point Insertion

- **Used to improve the Controllability and Observability of internal signal lines**



Control Test Point



Observe Test Point

Background: Test Point Insertion

- **Used to improve the Controllability and Observability of internal signal lines**
- **Control Test Points break signal path**
  - **Delay associated with multiplexer**
- **Controllability/Observability analysis used to identify potential test points**

Boolean Satisfiability (SAT)

- **Determining a satisfying variable assignment for a Boolean function**
- **Proof that no such variable assignment exists**
  - **Unsatisfiable (UNSAT)**
- **Expressed in Conjunctive Normal Form (CNF)**
  - **conjunction of clauses**
- **Clause – Disjunction of Literals**
- **Literal – Variable in positive or negative polarity**

CNF Formula

$$1.\quad (a+\overline{c})(\overline{b})(b+c)(\overline{b}+c)$$

$$SAT:\{b=0, c=1, a=1\}$$

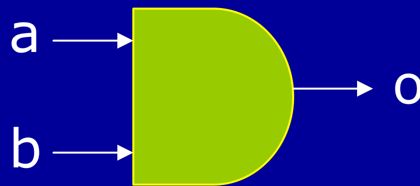
$$2.\quad (a+\overline{c})(\overline{b})(b+c)(b+\overline{c})(\overline{b}+c)$$

Unsatisfiable

UNSAT Core: Minimum set of clauses that cause the SAT instance to be Unsatisfiable

Transforming a Circuit to SAT

- **Any circuit can be transformed into a propositional formula**
- **For Example**



$$(a + \overline{o})(b + \overline{o})(\overline{a} + \overline{b} + o)$$

- **Test generation for a fault can also be transformed to a propositional formula**
  - **Larabee [TCAD, 1992]**

Boolean Satisfiability (SAT)

- To generate a test pattern for a single fault
  - Extract a formula defining set of patterns that detect the fault
  - Use SAT algorithm to satisfy the formula
- NP Complete Problem
- Recent advances lead to identification of UNSAT Core
  - Points to Untestable parts of circuit
- Learned Clauses (derived while solving SAT instance)
  - Identify most frequent parts of circuit

Proposed Scheme

- **Insertion of Test Points at RTL**
  - **To reduce tester storage**
  - **Transition testing**
- **ATPG, scan chain insertion etc at gate level**

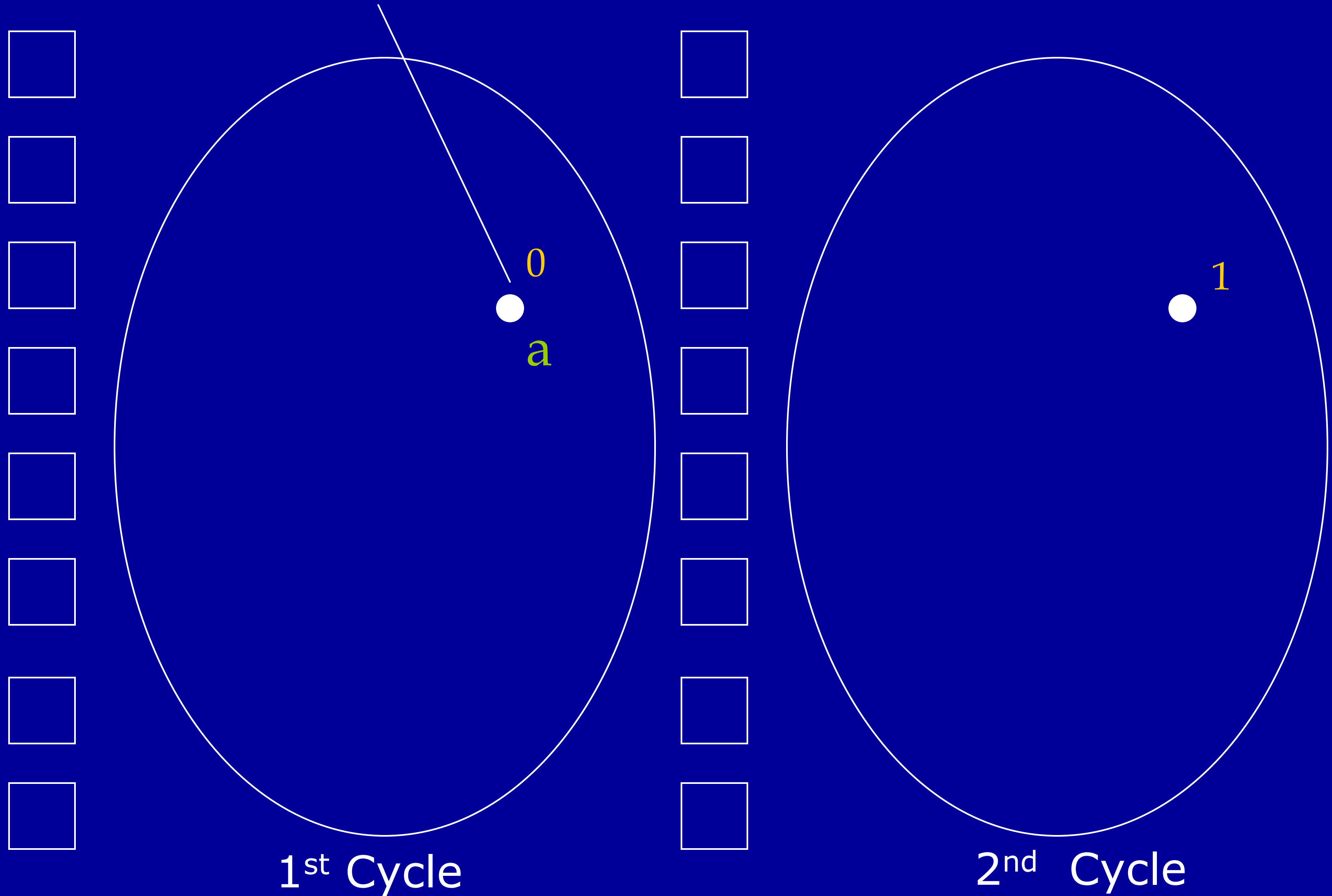
- **Advantage of TPI at RTL**
  - **Extra delay can be absorbed during logic synthesis**
- **Currently, test points in scan flip-flops only**
- **Launch-on-Capture (Broadside) testing**

Previous Work

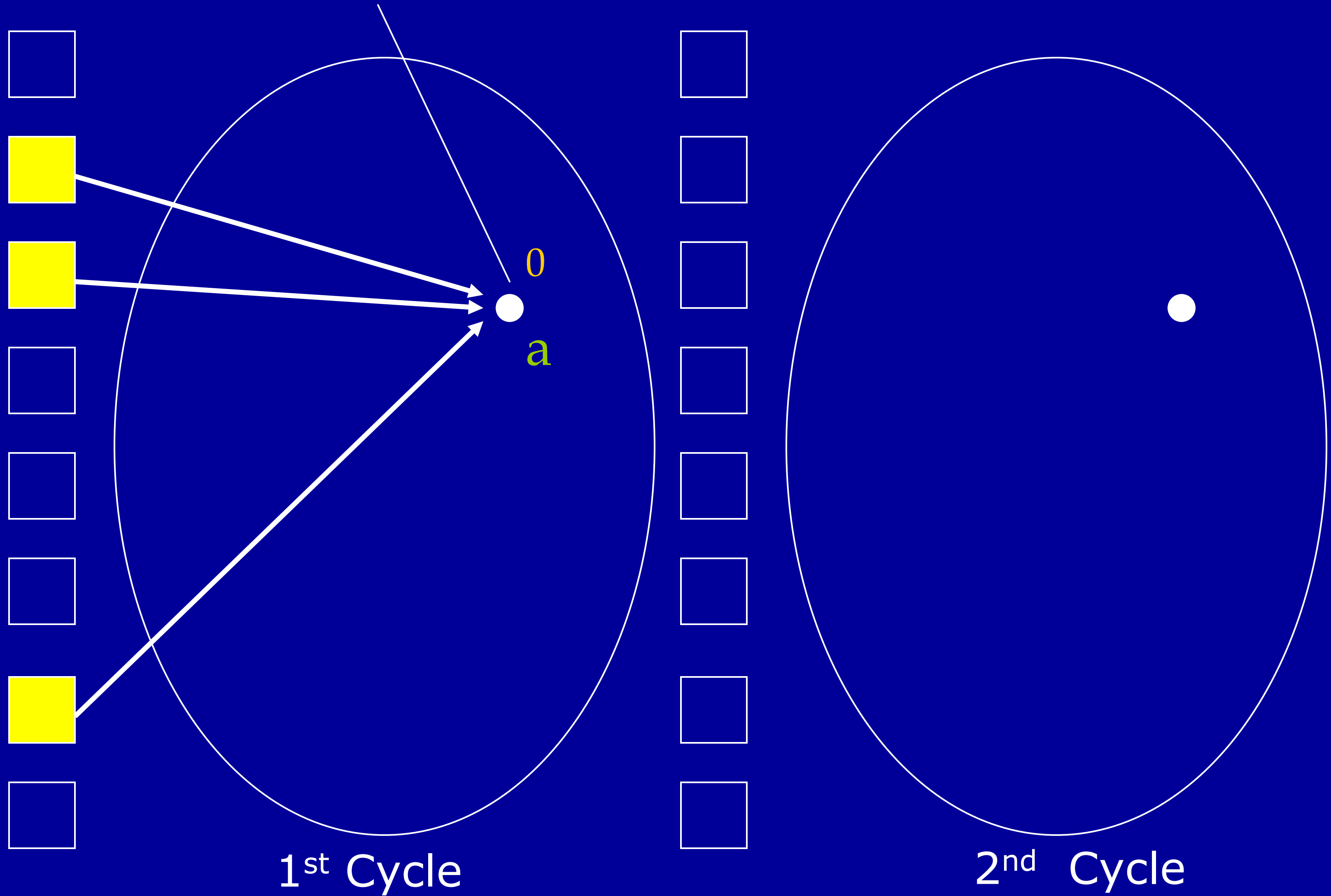
- **Test Point Insertion at RTL**
  - **Improving testability**
- **Testability analysis and test point insertion at RTL for scan based BIST**
  - **S. Boubezari [IEEE TCAD Sept. 1999]**
  - **J. Carletta and C. Papachristou [ICCD 95]**
  - **S. Roy, G. Guner and K.-T Cheng [ITC 2000]**
- **Unsatisfiability based non-scan DFT**
  - **L. Lingappan and N. Jha [VTS 2005]**
- **Proposed Scheme**
  - **Emphasis on test data volume, not coverage**

Key Idea

- Insert test points to reduce the specified bits required in transition fault testing
- Efficiency of test compaction and test compression
  - Directly proportional to the specified bits
- Reducing the dependency of the second pattern
- Scan flip-flops that require a lot of specified bits in the first time frame
  - Controlled directly

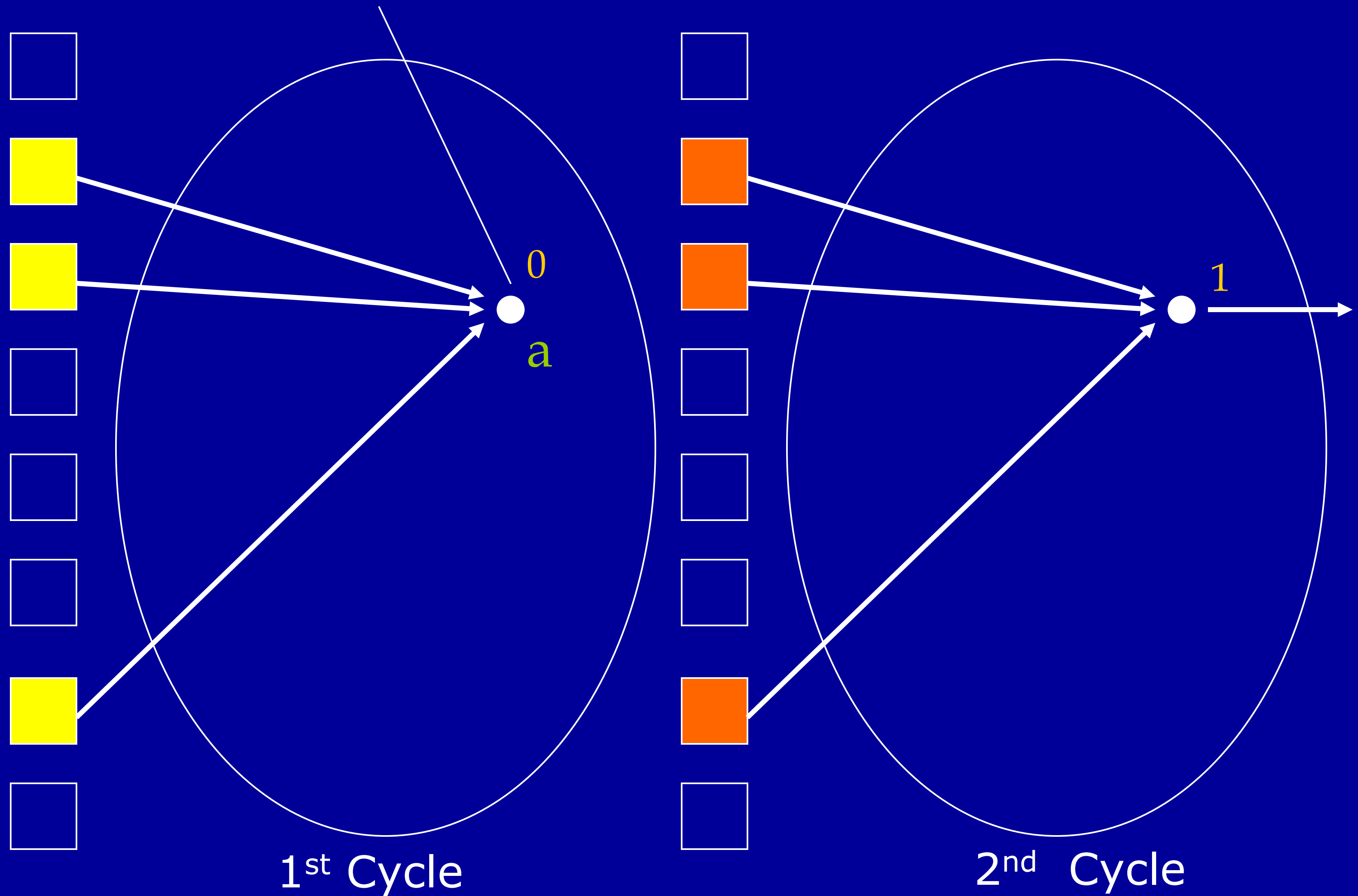


Generating broadside patterns at line a

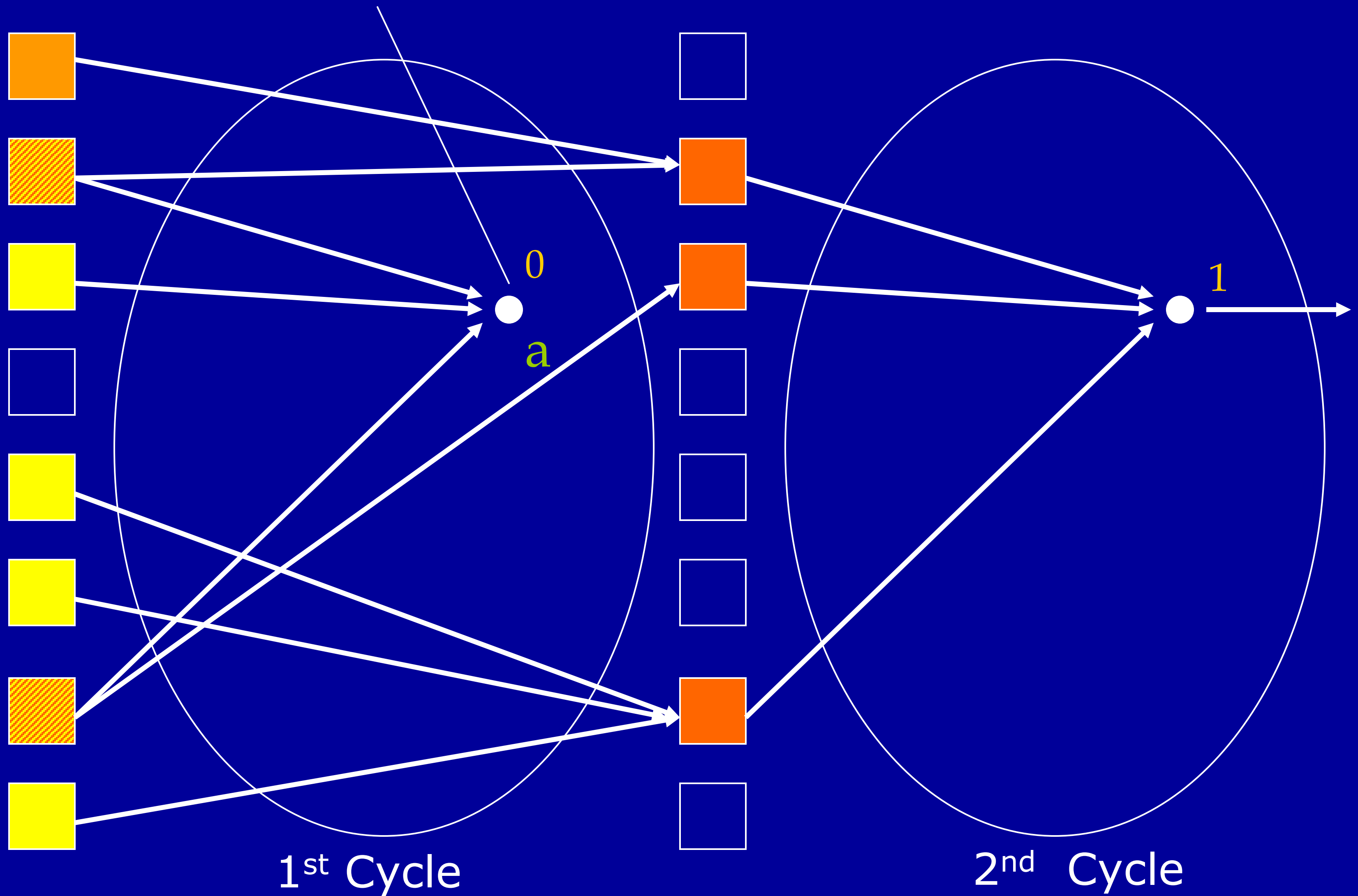


Need to specify flip-flops to set line to 0

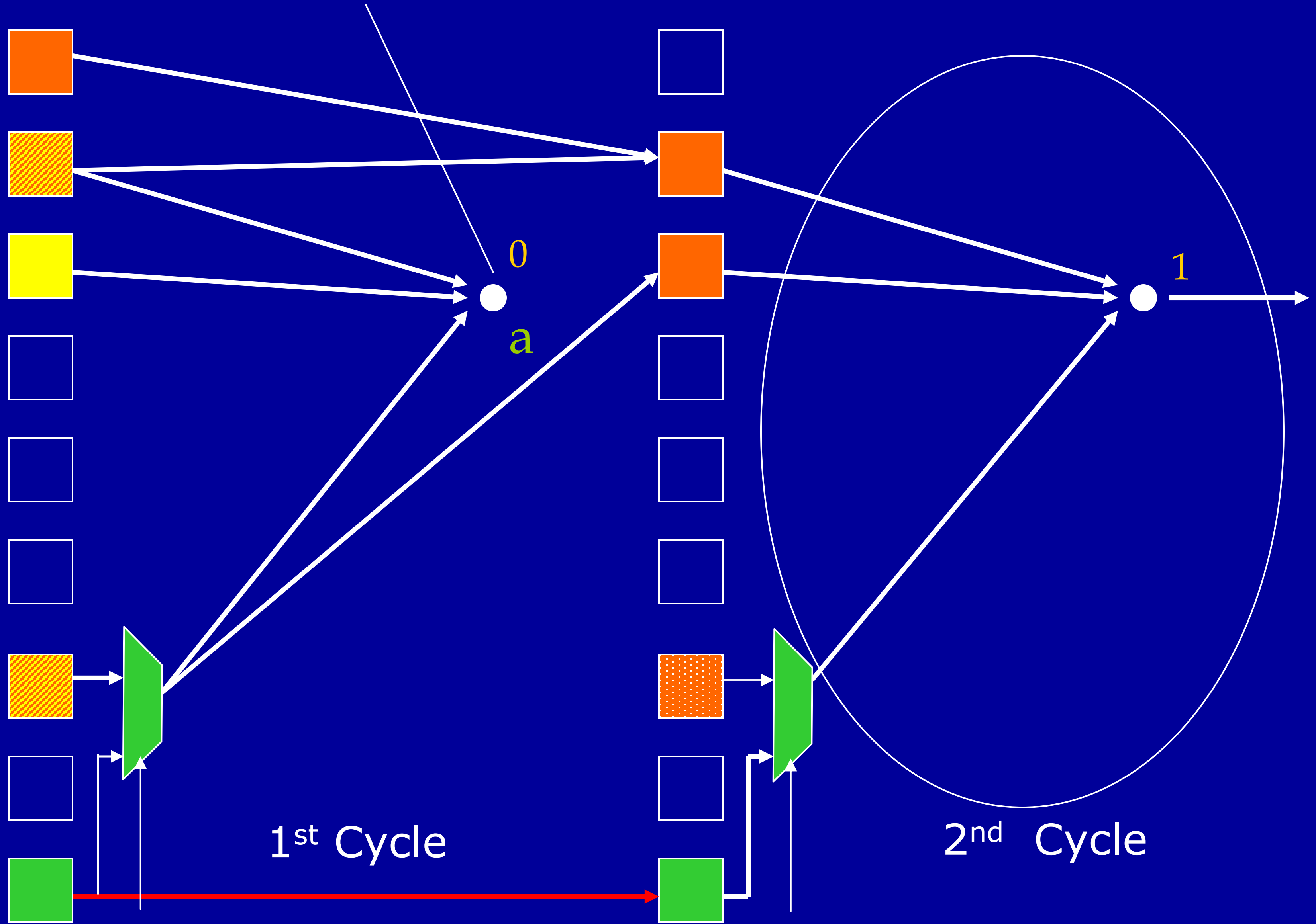
Transition fault (slow to rise)



To set line to 1, more FFs controlled in 2nd cycle



Broadside test requires lots of FFs to be specified



Additional Flip-Flop will have required value in 2nd cycle

Challenges of TPI at RTL

- **Structural information may not be available**
- **Depending on synthesis constraints, may map into different gate level netlists**
- **Technique should work across all synthesis constraints**
- **Need to use *functional* information**
- **Proposed technique is based on boolean satisfiability**

Our Approach

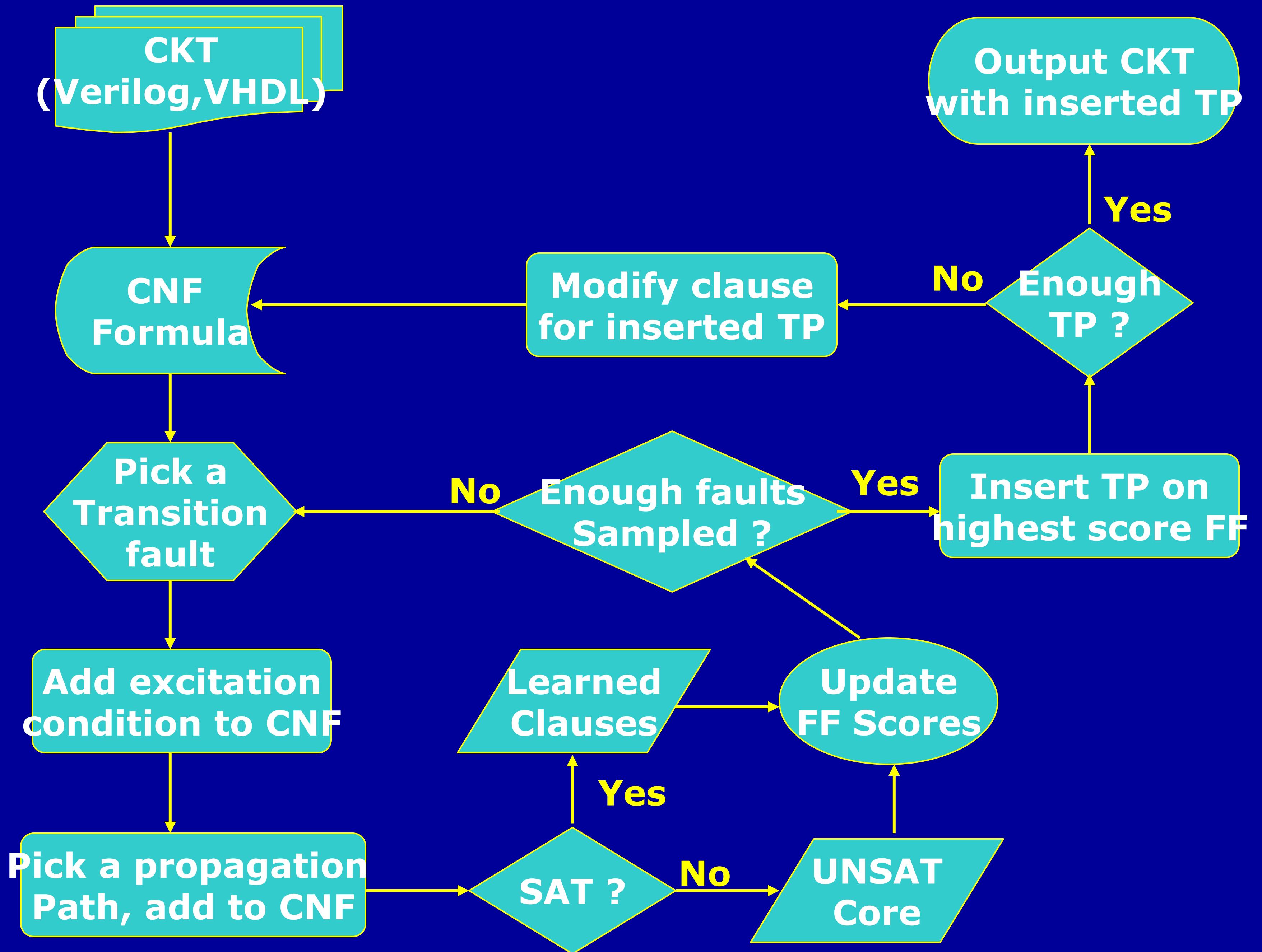
- **Test Points inserted only in scan flip-flops**
- **Identify flip-flops that are**
  - **Specified most number of times in transition test patterns**
  - **Difficult to control in the second time-frame**
- **Traditional controllability/observability techniques don't work well**
  - **Adequate structural information not available**
- **SAT based heuristic better**
  - **RTL primitives like adder, multiplexers can be transformed to CNF**

Identifying test points

- **Fast (and approximate) test generation using SAT**
  - **Subset of transition faults**
- **UNSAT Core and Learned Clauses**
  - **Used to identify potential candidates**
- **Maintain score for each flip-flop**
- **If SAT instance is UNSAT**
  - **Score of flip-flops appearing in UNSAT core increased**

Identifying test points

- **If SAT instance is Satisfiable**
  - **Score of flip-flops in learned clauses are increased**
- **Scan flip-flop with highest score after sampling**
  - **Identified for test point insertion**
- **Weighted scores can be used for UNSAT and Learned clauses**
  - **Weights depending on circuit, test generation ease**


CKT
(Verilog,VHDL)
Output CKT
with inserted TP
Yes
CNF
Formula
Modify clause
for inserted TP
No
Enough
TP ?
Pick a
Transition
fault
No
Enough faults
Sampled ?
Yes
Insert TP on
highest score FF
Add excitation
condition to CNF
Learned
Clauses
Update
FF Scores
Yes
Pick a propagation
Path, add to CNF
SAT ?
No
UNSAT
Core

Heuristics

- Fault sampling
  - Initially, faults chosen randomly
  - Faults are selected on lines with lowest scores
- Score maintained for each line
  - Number of times it has occurred before
- Propagation path
  - Selected using a greedy heuristic
  - Next node selected as the node in the fan-out that has minimum score
  - Till the path reaches primary output or flip-flop

Implementation

- **Test point selection algorithm implemented in C++**
- **Integrated with ZCHAFF [Zhang and Malik, DATE 2003] Boolean SAT solver**
- **Behavior level circuit**
  - **Read using Verific HDL parser**
  - **Stored as temporary RTL netlist**

Experimental Setup

- **4 ITC'99 benchmarks and 4 SoC Cores**
  - **Assumed to be full-scan**
- **Insert Test Points in 2 % flip-flops using the proposed technique**
- **Use the same delay constraints for logic synthesis**
- **'report_area' command of Design Compiler to get the area**
- **Proprietary ATPG to generate transition patterns**

Experimental Results: Benchmarks

CKT	# FF	Vect	FC %	FE %	ATPG (s)	Clk (ns)	Area
b14	250	622	66.98	92.68	459.2	12.68	36157
	256	347	67.04	89.67	73.9	12.64	36565
b15	460	935	58.35	85.61	1899.1	9.9	56721
	470	395	58.41	85.64	298.3	9.8	57513
b20	501	891	69.96	93.51	947.0	12.77	76952
	512	636	70.00	92.63	244.6	12.53	77728
b21	497	907	70.64	93.88	984.8	12.83	76866
	508	520	70.72	86.92	566.1	12.61	78137

Experimental Results: SoC Cores

CKT	# FF	Vect	FC %	FE %	ATPG (s)	Clk (ns)	Area
D1	337	1703	85.48	98.94	364.5	7.51	33548
	345	1453	85.50	98.45	294.9	7.84	33535
D2	876	1166	69.16	93.56	906.9	9.81	60252
	895	691	69.17	92.03	290.8	9.77	61623
D3	1858	2347	58.66	90.87	3216.9	9.84	105174
	1897	1968	58.66	92.45	1656.4	9.68	105440
D4	3738	1688	34.76	92.32	5422.2	10.94	250049
	3863	1286	34.76	92.67	2702.1	10.94	251619

Experimental Results

CKT	Volume Red. (%)	ATPG run Red. (%)	Area Overhead	Run Time (s)
b14	42.87	83.91	1.13 %	248.9
b15	56.84	84.29	1.40 %	182.9
b20	27.05	74.17	1.01 %	245.1
b21	41.40	42.52	1.65 %	484.2
D1	12.65	19.09	-0.04 %	157.8
D2	39.45	67.93	2.28 %	422.9
D3	14.39	48.51	0.25 %	378.0
D4	21.85	50.17	0.63 %	2720.5
Avg.	32.06	58.82	1.03 %	

Conclusions and Future Work

- **Methodology to insert test points at RTL**
  - **Aimed at transition tests**
- **Reducing the specified bits required in transition faults**
  - **Improving test compaction/compression**
- **Test Data Volume reduced by 32%**
  - **Around 1% area overhead**
- **Enhance the fault sampling/propagation path selection heuristic**
- **Define a good metric to measure the impact on test compression**

Questions