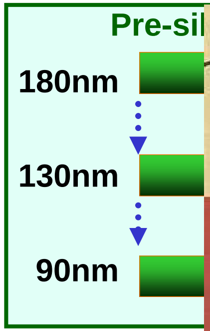
Fixing Design Errors with Counterexamples and Resynthesis

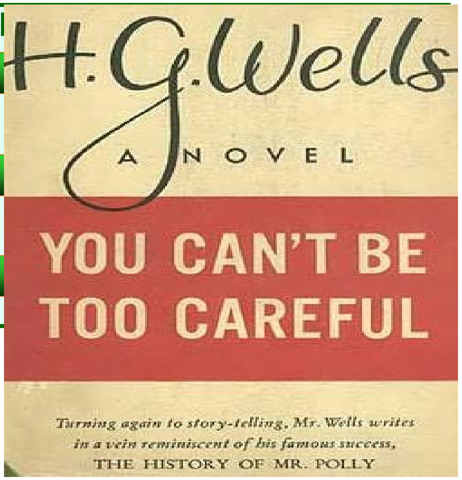
Kai-hui Chang,
Igor L. Markov and Valeria Bertacco

University of Michigan at Ann Arbor

Jan. 26, 2007

Current Design Challenges

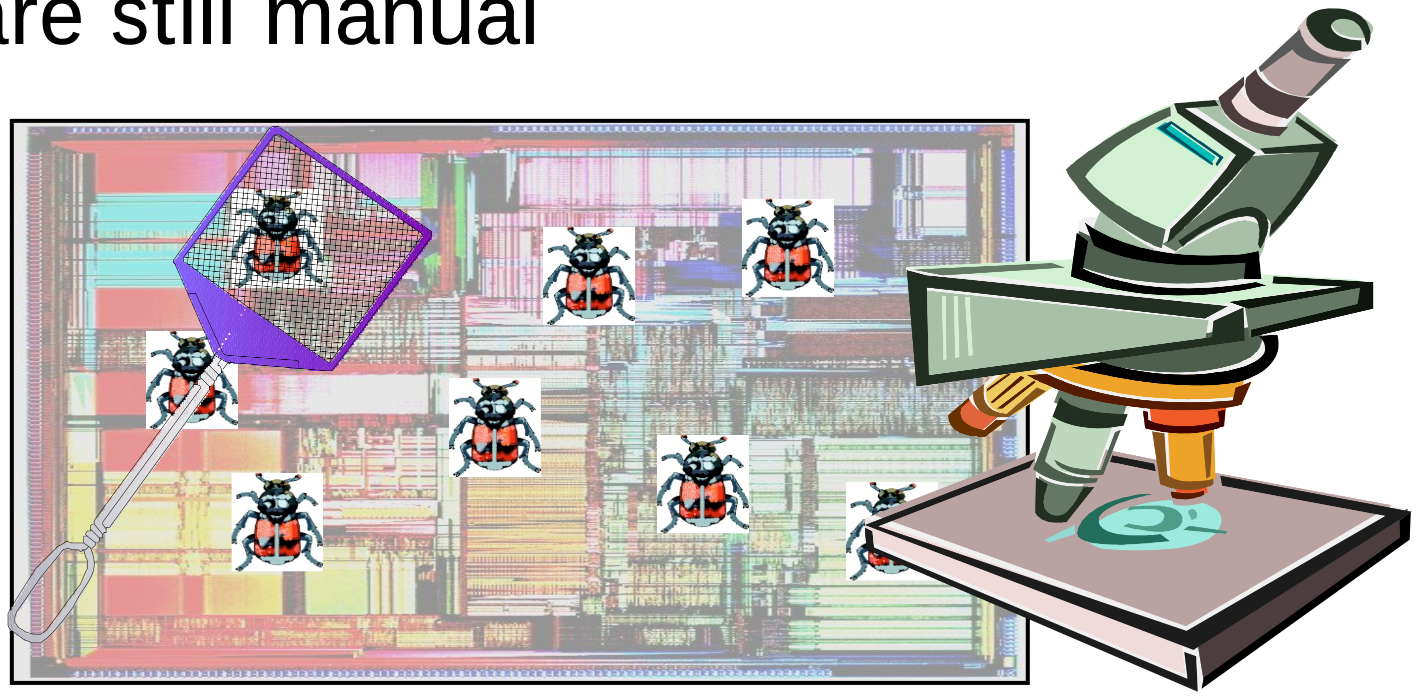




- Explosive design complexity – verification becomes more difficult
  - 50% of the designs will have functional mistakes at the first tape-out
- Verification limits the features that can be implemented in a design [Chayut'06]
- Decreased time to market → shorter verification time
- Respin is expensive
  - Mask cost is approaching $1 million per set

Current Trends

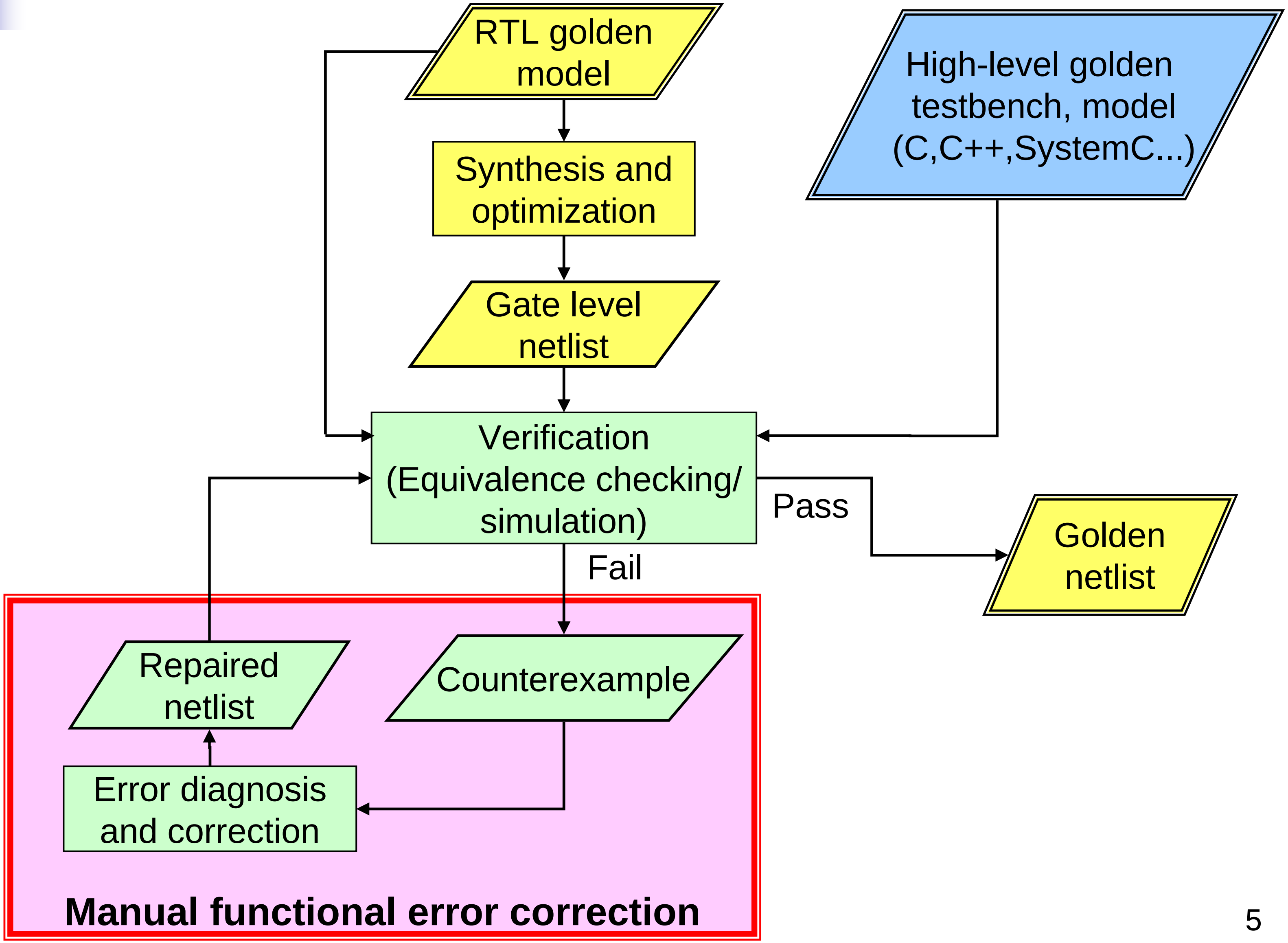
- Testbench generation and verification have been automated
- Error diagnosis and correction are still manual



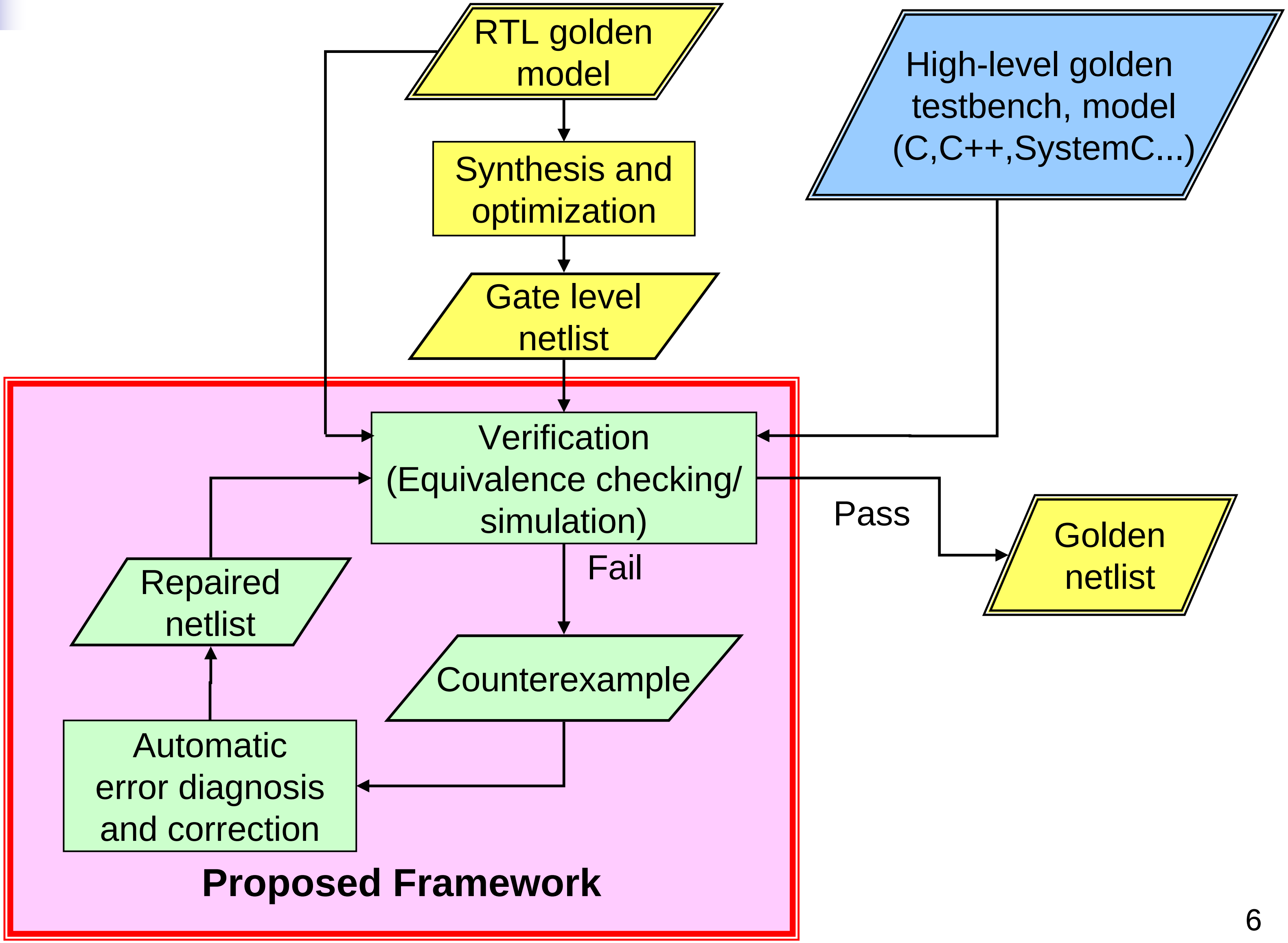
Current Trends

- Testbench generation and verification have been automated
- Error diagnosis and correction are still manual
- Diagnosing and correcting design bugs are especially difficult at the gate level
  - Engineers unfamiliar with synthesized netlists
  - Bug fixing is difficult, time-consuming

Current Back End Logic Design Flow

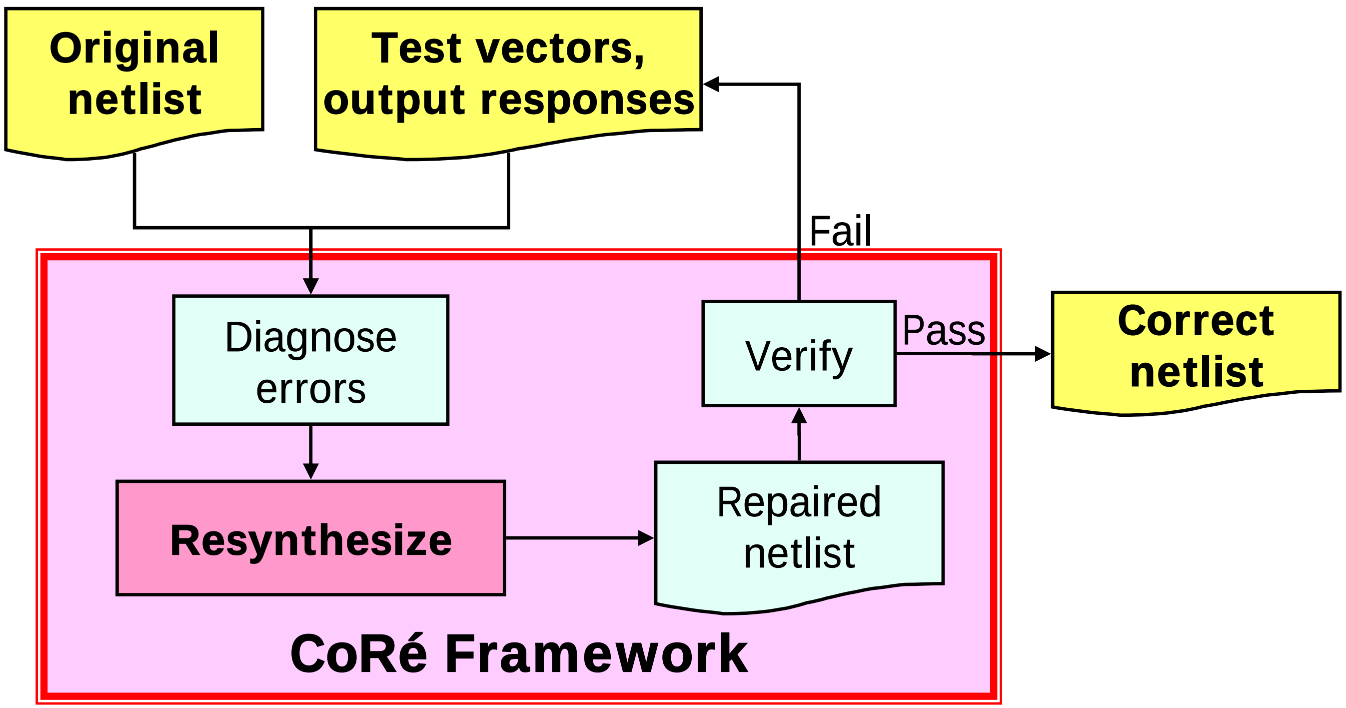


Proposed Back End Logic Design Flow



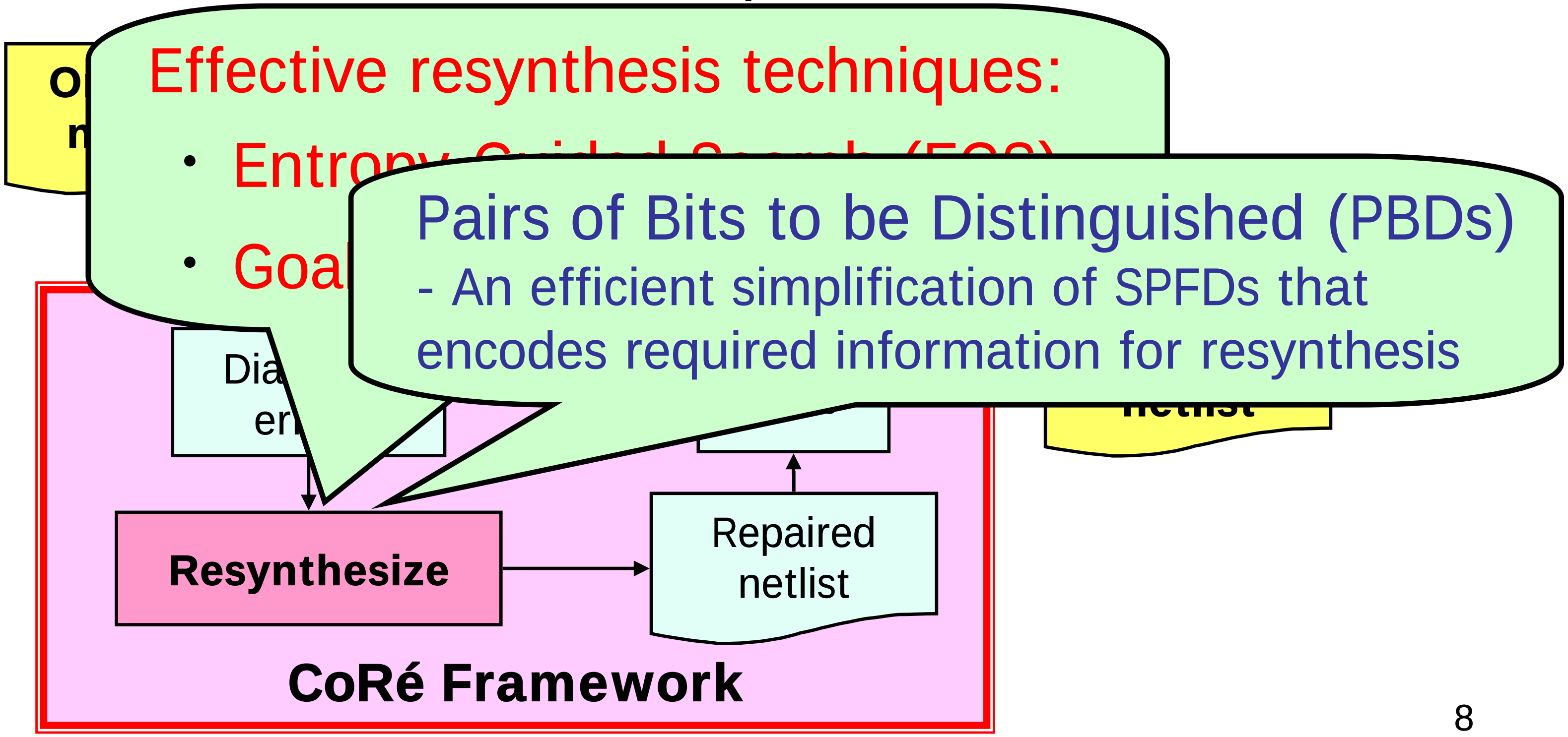
Contributions

- COunterexample-guided REsynthesis framework (CoRé) for combinational circuits
  - Abstraction: signatures produced by simulation
  - Refinement: counterexamples that fail verification



Contributions

- COunterexample-guided REsynthesis framework (CoRé) for combinational circuits
  - Abstraction: signatures produced by simulation
  - Refinement: counterexamples that fail verification



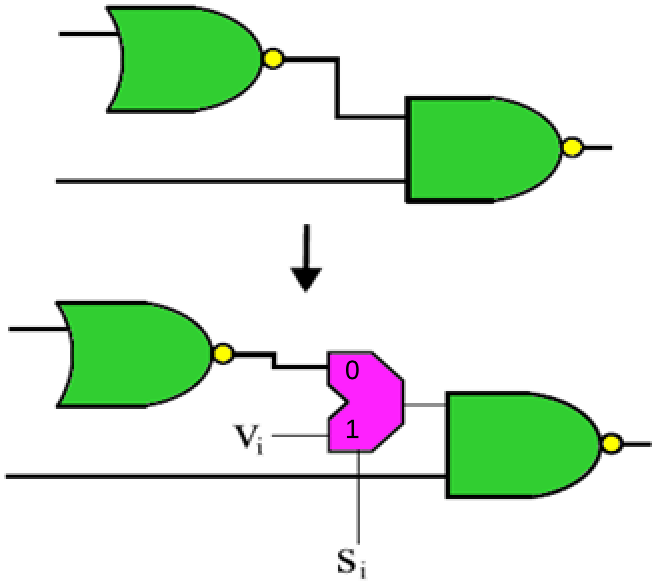
Outline

- CoRé Framework
- Resynthesis techniques
  - Entropy-Guided Search (EGS)
  - Goal-Directed Search (GDS)
- Previous work
- Experimental results
- Conclusions

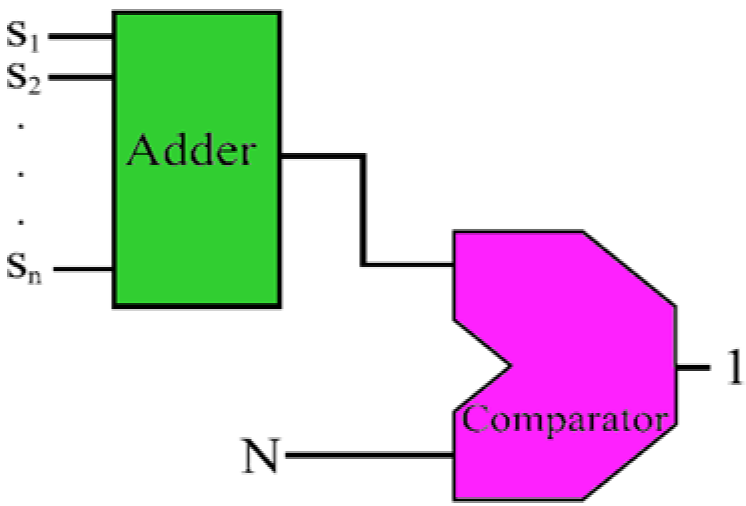
Error Diagnosis

[Smith *et al.*, ASPDAC'04]

1. To model errors: insert MUXes into the circuit
2. To limit the number of allowed errors: use an adder and a comparator
3. Convert the circuit to CNF
4. Constrain inputs/outputs using input vectors/correct output responses



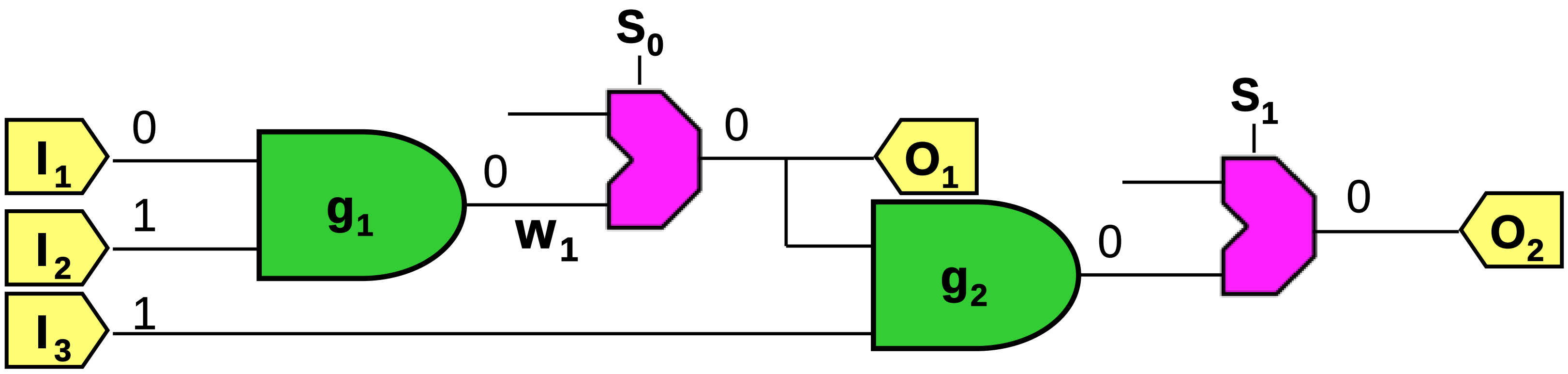
Error modeling



Error-cardinality constraint

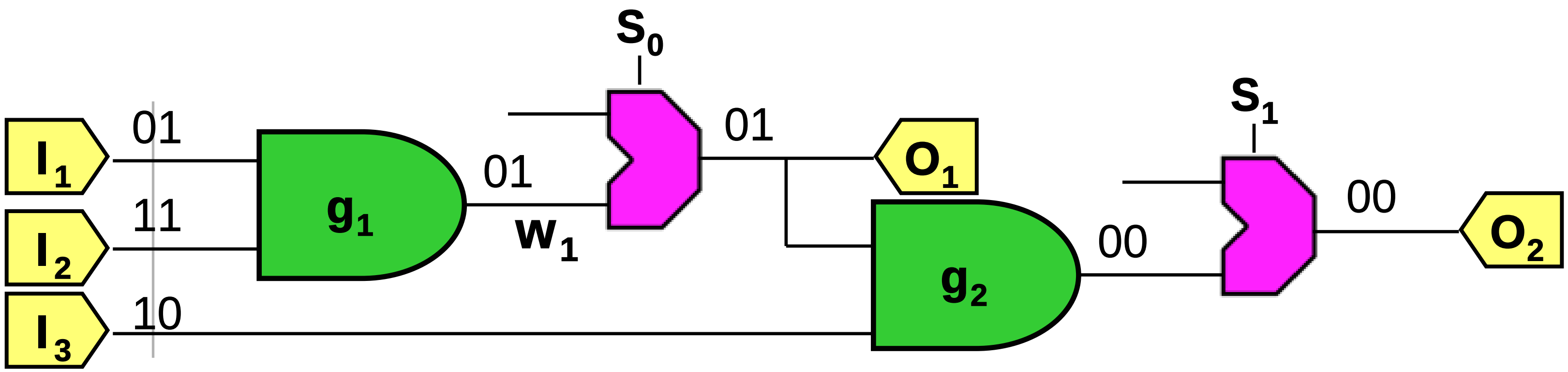
CoRé Framework

- Simulate bug traces to generate signatures



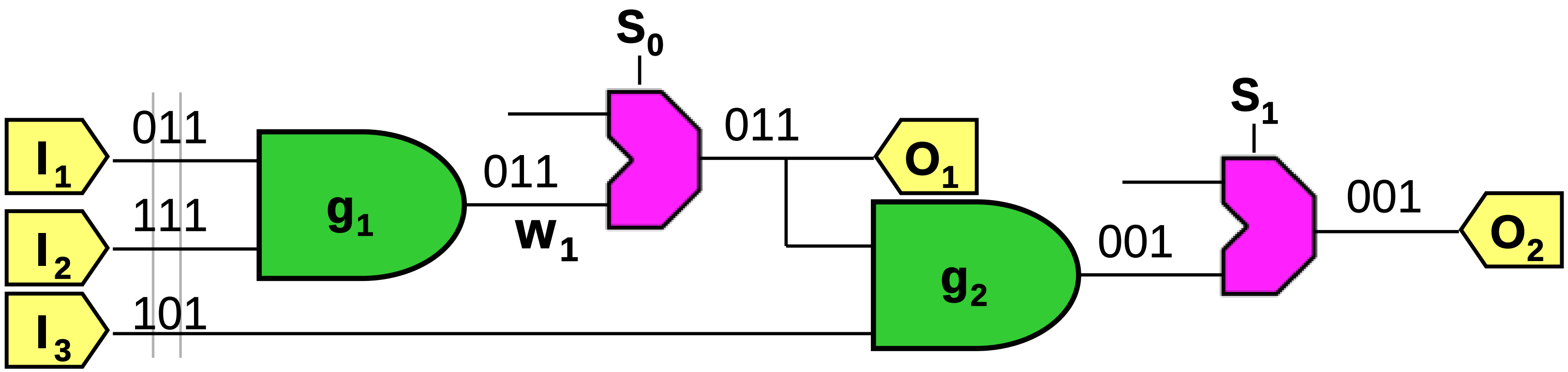
CoRé Framework

- Simulate bug traces to generate signatures



CoRé Framework

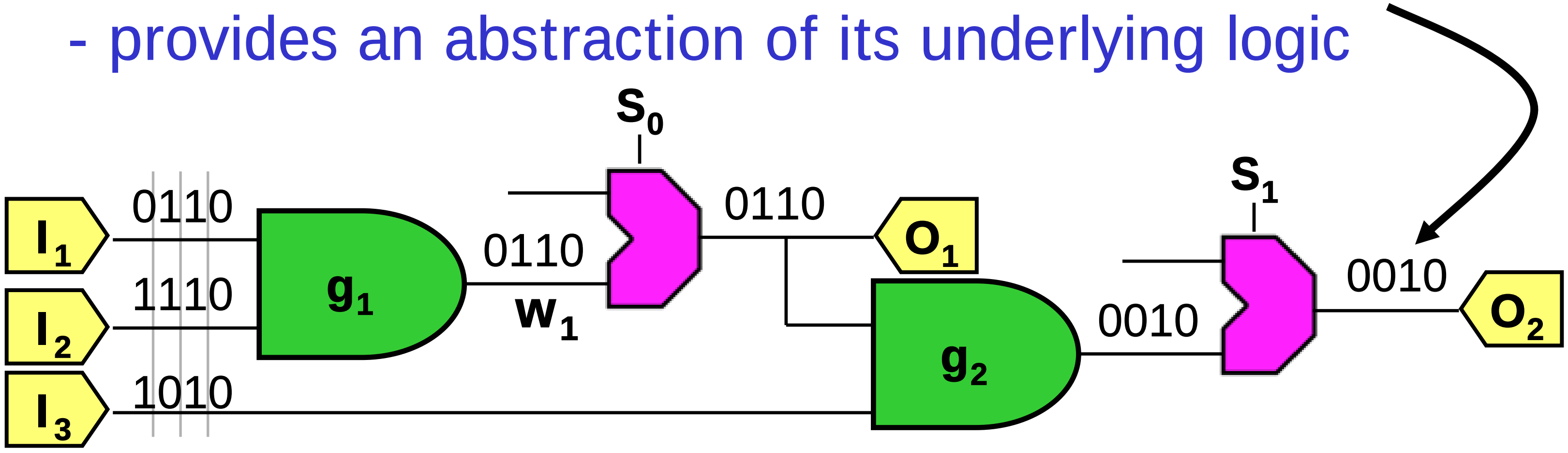
- Simulate bug traces to generate signatures



CoRé Framework

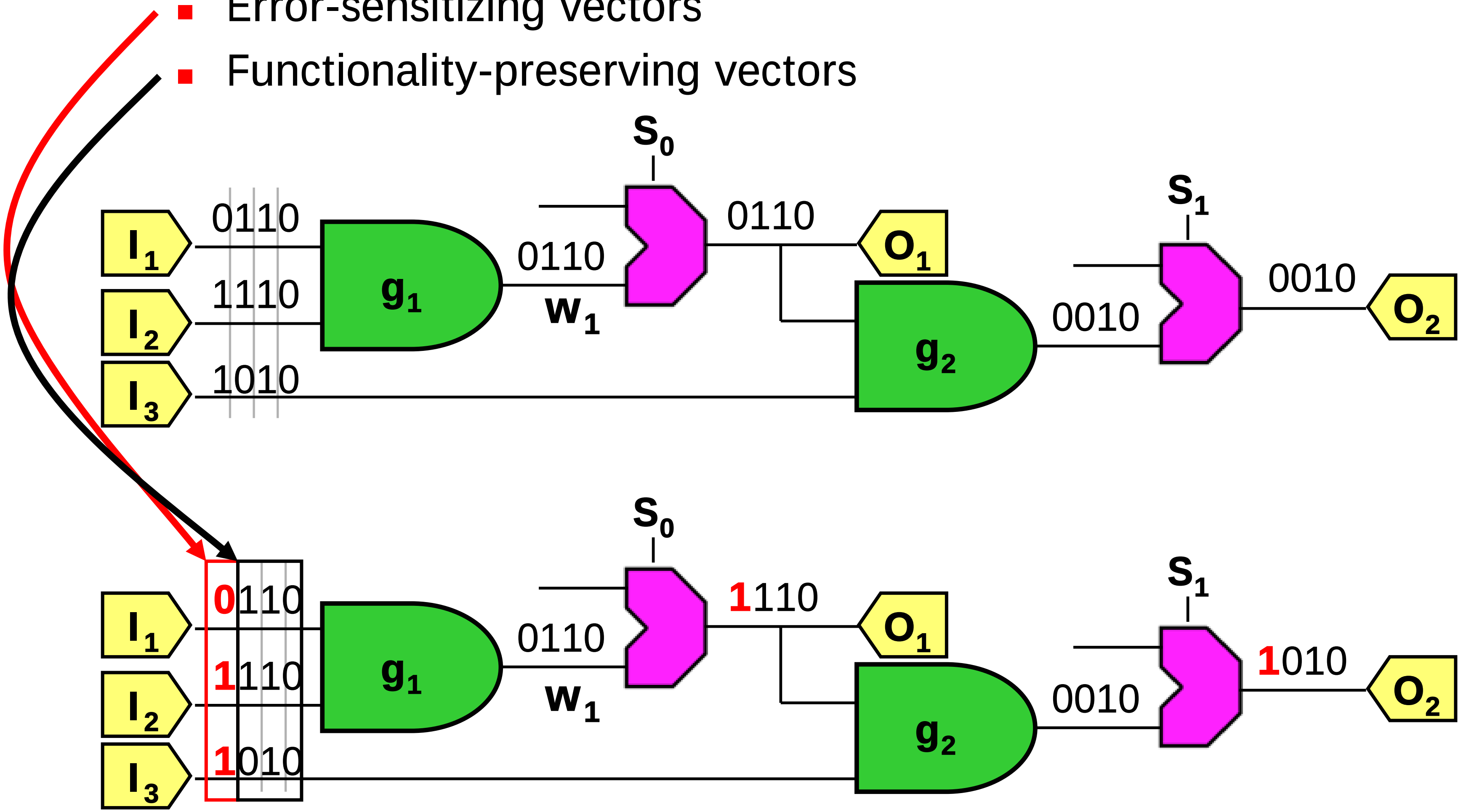
- Simulate bug traces to generate signatures

  A signature of a signal is its partial truth-table - provides an abstraction of its underlying logic



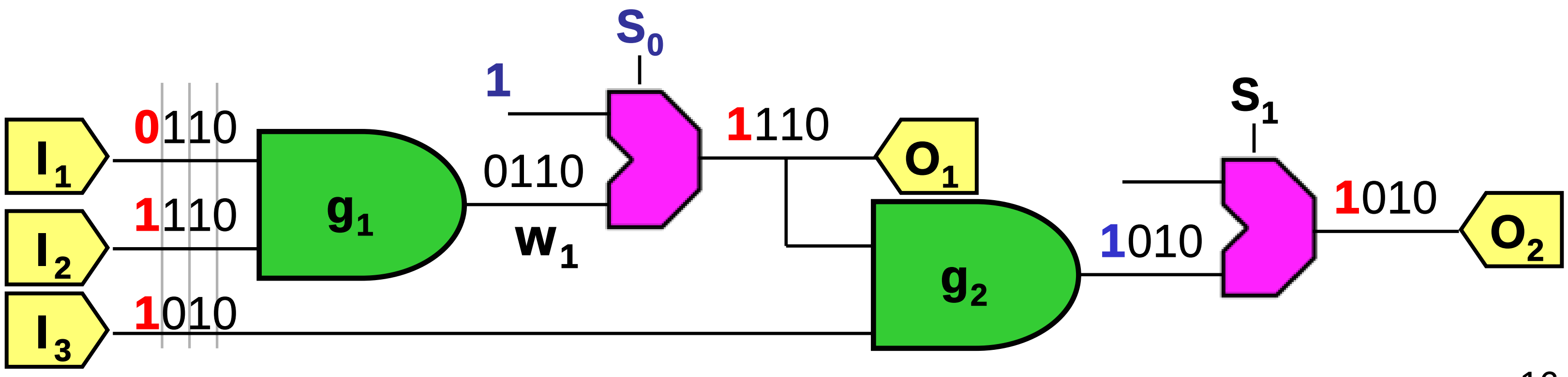
CoRé Framework

- Simulate bug traces to generate signatures
  - Error-sensitizing vectors
  - Functionality-preserving vectors



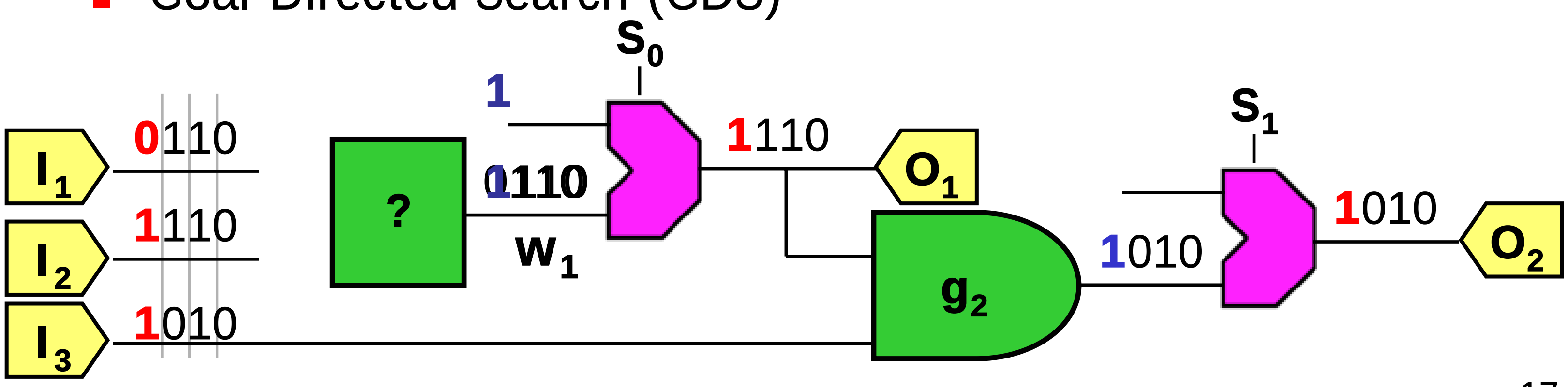
CoRé Framework

- Simulate bug traces to generate signatures
  - Error-sensitizing vectors
  - Functionality-preserving vectors
- Perform error diagnosis using error-sensitizing vectors
  - Error sites and values to correct outputs of error-sensitizing vectors are returned



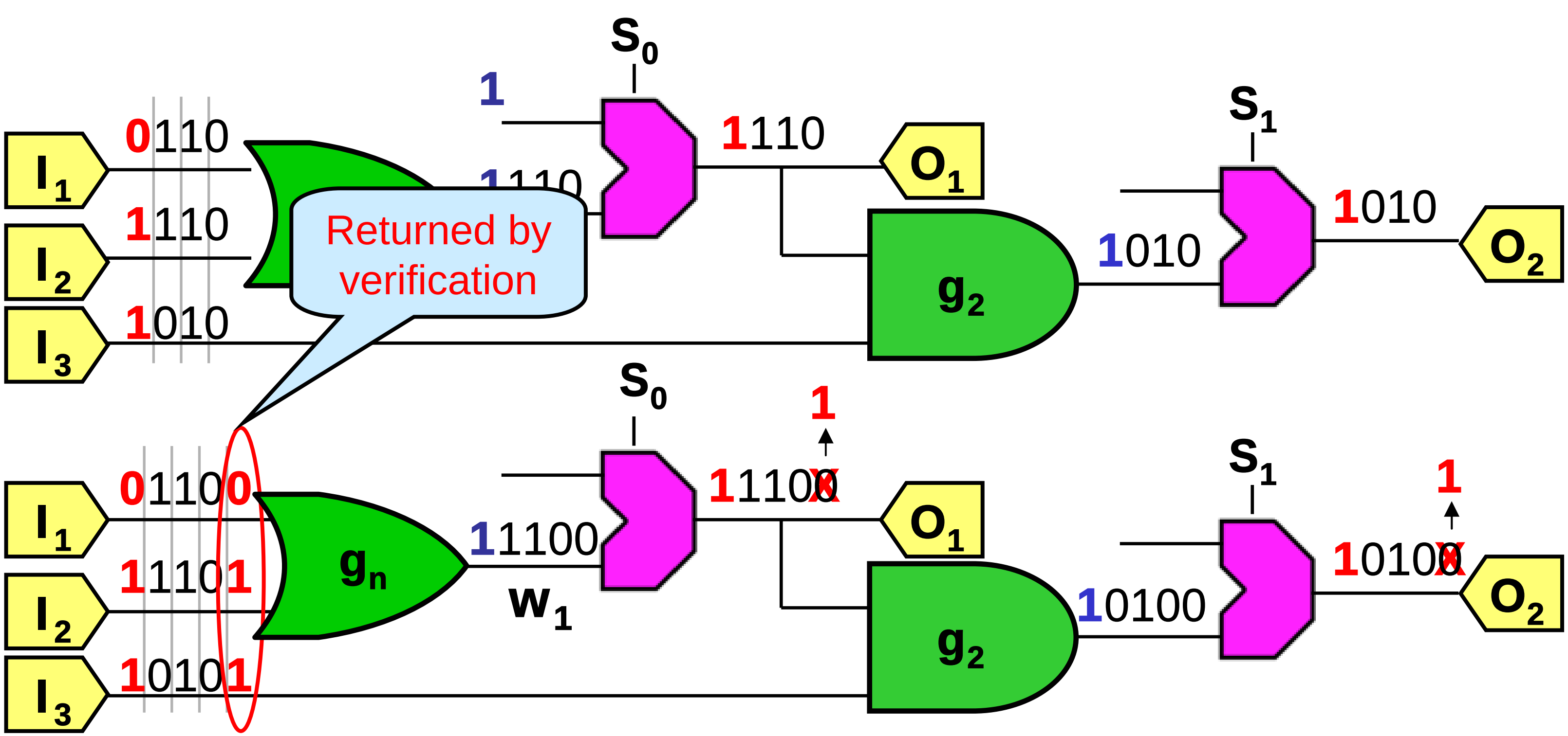
CoRé Framework

- Simulate bug traces to generate signatures
  - Error-sensitizing vectors
  - Functionality-preserving vectors
- Perform error diagnosis using error-sensitizing vectors
  - Error sites and values to correct outputs of error-sensitizing vectors are returned
- Resynthesize the error site using the signature
  - Entropy-Guided Search (EGS)
  - Goal-Directed Search (GDS)



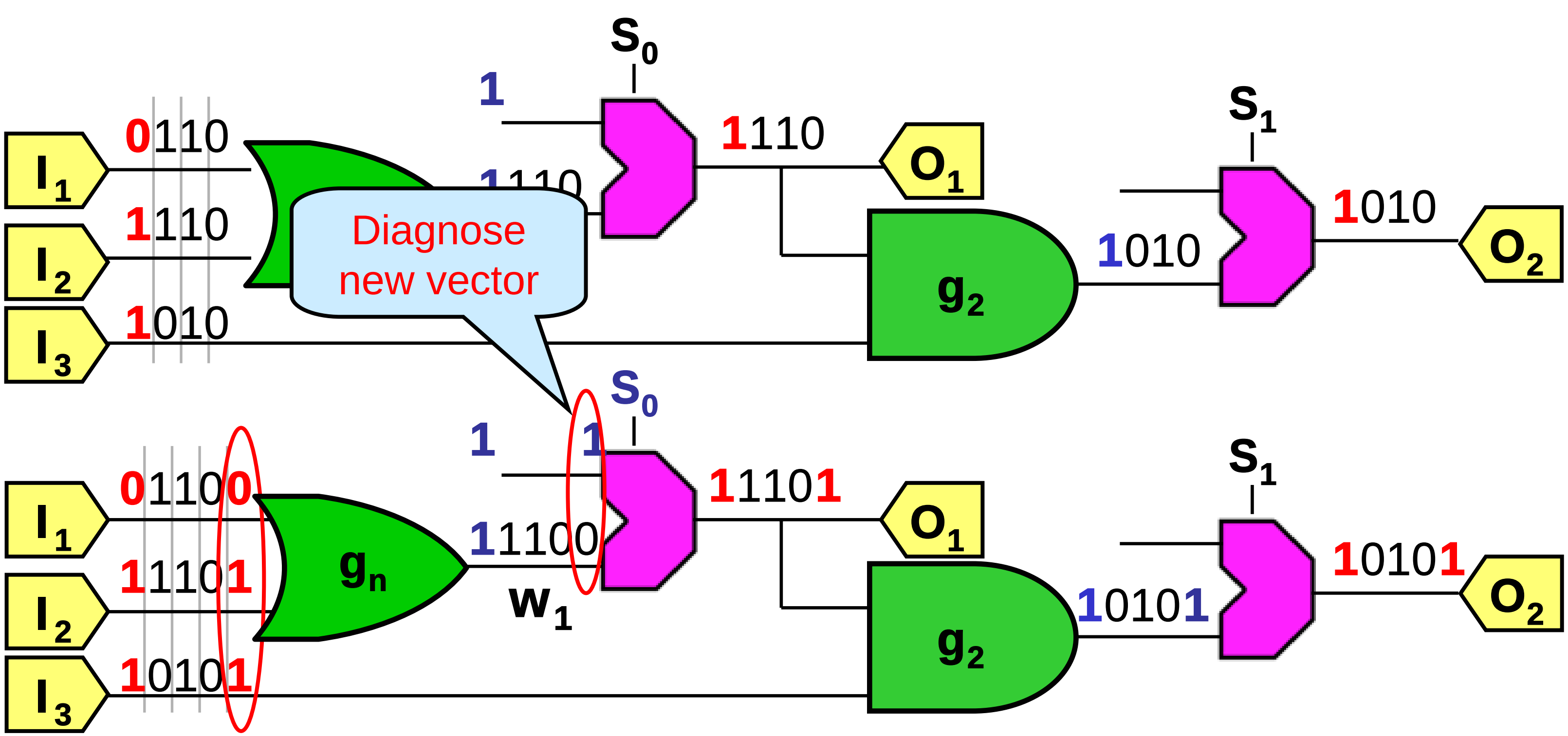
CoRé Framework

- Refinement of abstraction
  - If the fix is incorrect, new bug traces will be used to refine the signatures



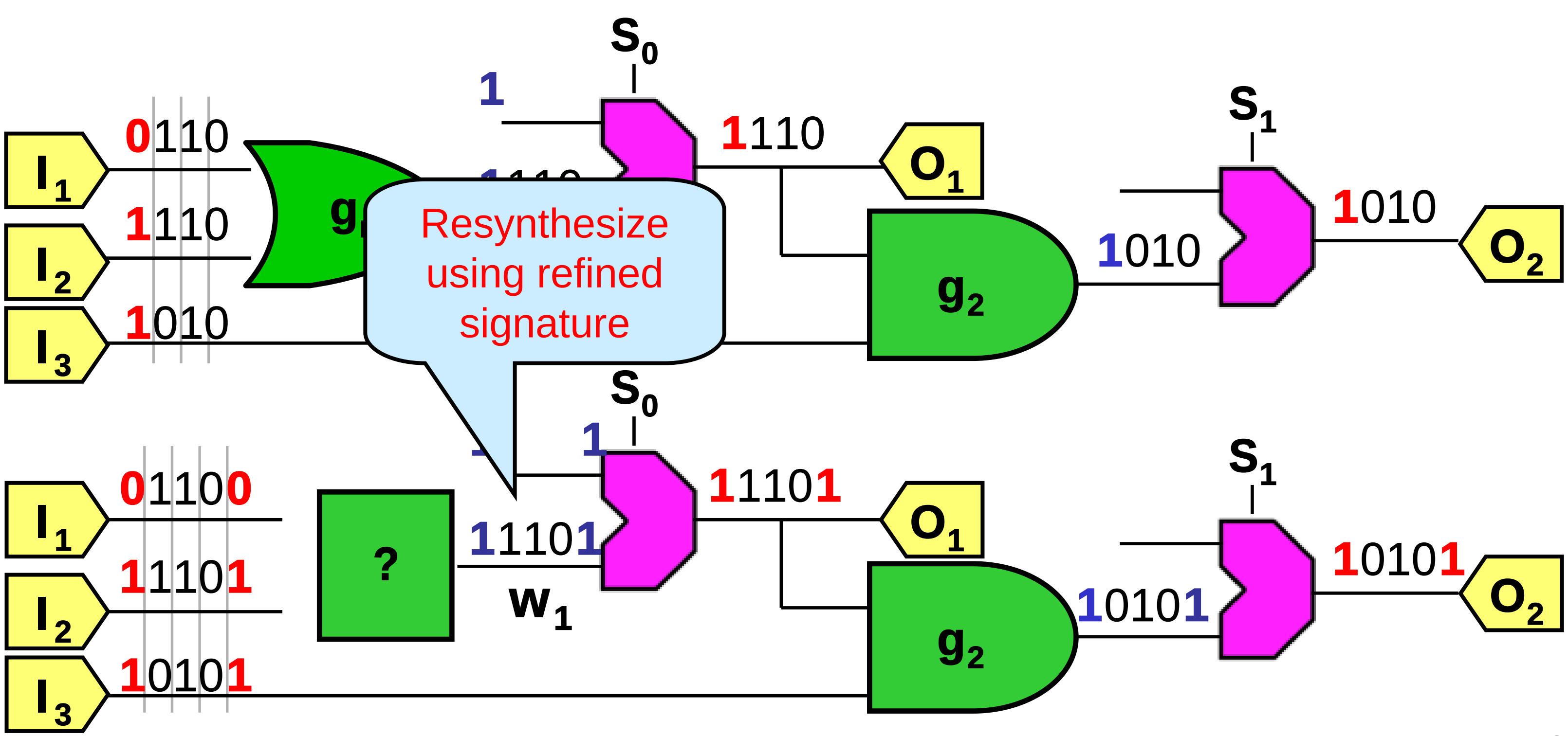
CoRé Framework

- Refinement of abstraction
  - If the fix is incorrect, new bug traces will be used to refine the signatures



CoRé Framework

- Refinement of abstraction
  - If the fix is incorrect, new bug traces will be used to refine the signatures

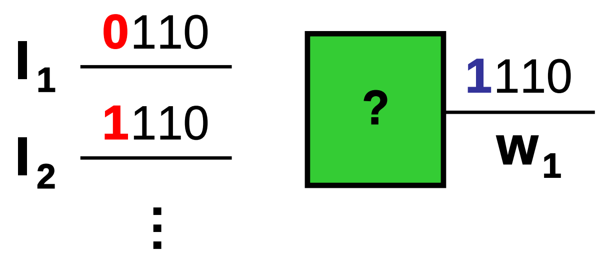


Outline

- CoRé Framework
- Resynthesis techniques
  - Entropy-Guided Search (EGS)
  - Goal-Directed Search (GDS)
- Previous work
- Experimental results
- Conclusions

The Resynthesis Problem

- Problem formulation
  - Given a target signature
  - Find a resynthesis netlist that generates the target signature using other signatures



- How to find input signatures that can generate the target signature?
- How to find a resynthesis netlist using the input signatures?

Selecting Input Signatures

Target signature (s_t)	0 0 1 1 1
Candidate signature 1 (s_{c1})	1 0 1 1 0
Candidate signature 2 (s_{c2})	1 1 1 1 0
Bit index	1 2 3 4 5

{1, 3}

- Target signature cannot be generated using these two signatures
  - Values of bits {1, 3} are different in the target signature but are the same in all candidate signatures

Selecting Input Signatures

Target signature (s_t)	0 0 1 1 1
Candidate signature 1 (s_{c1})	1 0 1 1 0
Candidate signature 2 (s_{c2})	0 1 1 1 0
Bit index	1 2 3 4 5

- The target signature can be generated using the candidate signatures
  - $s_t = \overline{s_{c1} \oplus s_{c2}}$
- For any pair of bits in the target signature whose values are different
  - The corresponding bits in candidate signatures are never the same

Selecting Input Signatures

Target signature (s_t)	0	0	1	1	1
Candidate signature 1 (s_{c1})	1	0	1	1	0

Theorem 1 [Zhang, IWLS'05]

Consider candidate signatures $s_{c1}, s_{c2}, \ldots, s_{cn}$
and a target signature s_t
Then a resynthesis function F, where
$s_t = F(s_{c1}, s_{c2}, \ldots, s_{cn})$, exists
if and only if no bit pair $\{i, j\}$ exists such that
$s_t[i] \neq s_t[j]$ but $s_{ck}[i] = s_{ck}[j]$ for all $1 \leq k \leq n$.

- whose values are different
  - The corresponding bits in candidate signatures are never the same

Pairs of Bits to be Distinguished

Target signature (s_t)	0 0 1 1 1
Candidate signature 1 (s_{c1})	1 0 1 1 0
Candidate signature 2 (s_{c2})	0 1 1
Bit index	1 2 3 4 5

A PBD

Distinguished by s_{c2}

- Pair of Bits to be Distinguished (PB
  - A pair of bits in the target signature, indexed $\{i, j\}$, whose values are different
  - A PBD can be *distinguished* by a candidate signature s_{ck} if $s_{ck}[i] \neq s_{ck}[j]$
- For a resynthesis netlist to exist, all the PBDs in the target signature must be distinguished
  - This is a necessary and sufficient condition

Entropy of a Signature

Target signature s_t: $\underbrace{0000000}_{x \text{ 0s}}\ \underbrace{1111111111}_{y \text{ 1s}}$

- **Entropy of a signature** : $x \times y$
  (number of PBDs in the target signature)

Candidate signature s_c: $\overbrace{\underbrace{0011010}_{p \text{ 0s } q \text{ 1s}}}^{x \text{ bits}}\ \overbrace{\underbrace{1101010111}_{r \text{ 0s } s \text{ 1s}}}^{y \text{ bits}}$

- **Projected entropy of s_c w.r.t. s_t** :
  $p \times s + q \times r$ (number of PBDs distinguished by s_c)

Note: To simplify book keeping, bits in all signatures are rearranged so that the target signature resembles "0…01…1"

Entropy - Example

Signature	s_t	s_{c1}	s_{c2}	s_{c3}	s_{c4}
Pattern	00111	01011	10110	00101	00001
Entropy	6	3	3	4	2

- s_t can be generated using s_{c1}, s_{c2}, s_{c3}
  - All PBDs can be distinguished
  - Resynthesis function is $s_t = s_{c1} \,\&\, s_{c2} \mid s_{c3}$
- s_t cannot be generated using s_{c1}, s_{c4}
  - Not all PBDs can be distinguished
  - $SignatureEntropy(s_t) < PE(s_{c1}) + PE(s_{c4})$

Use of Entropy

- Theorem2
  - Consider a set of candidate signatures s_{c1} , s_{c2} ,…,s_{cn} and a target signature s_t
  - If s_t can be generated by s_{c1} , s_{c2} ,…,s_{cn} then $SignatureEntropy(s_t) \leq \sum PE(s_{ci})$
- A necessary, but not a sufficient condition

Entropy-Guided Search

- PBDs are used to select candidate signatures
  - Signatures that cover least-covered PBDs
  - Signatures with high entropy
  - Signatures that cover any uncovered PBDs
- A truth table is built using the selected signatures
  - Minterms not in the table are don't-cares
- The truth table can be synthesized by existing logic synthesis tools

EGS Example

Signature	Truth table				
S_t=1110	I_1	I_2	I_3	I_4	S_t
I_1=0110	0	1	1	0	1
I_2=1110	1	1	1	1	1
I_3=1101	1	1	0	0	1
I_4=0100	0	0	1	0	0
Synthesized	0	0	-	-	0

- Resynthesis function: $s_t = I_1 \mid I_2$
- The function is not unique

Outline

- CoRé Framework
- Resynthesis techniques
  - Entropy-Guided Search (EGS)
  - Goal-Directed Search (GDS)
- Previous work
- Experimental results
- Conclusions

Goal-Directed Search

- Recursively searches for valid resynthesis options
  - Branches using different gate types
  - Considers combinations of different inputs
- Efficient pruning techniques
  - Controlling values of logic gates
  - Entropy test – Theorem 2

Outline

- CoRé Framework
- Resynthesis techniques
  - Entropy-Guided Search (EGS)
  - Goal-Directed Search (GDS)
- Previous work
- Experimental results
- Conclusions

Previous Work

Technique	ED/ EC	Num. of Errors	Error model	Scalability	Requirement
ACCORD	Both	Single	SLDE	Moderate (BDDs)	Func. spec.
AutoFix	Both	Multiple	None	Moderate (BDDs)	Golden netlist
ICCAD'89	Both	Multiple	None	Moderate (BDDs)	Golden netlist
PRIAM	Both	Single	PRIAM	Moderate	Func. spec.
CHARME'05	Both	Multiple	None	Moderate	Func. spec.
EDAC'92	ED	Single	Abadir	Good (ATPG)	Test vectors
TCAD'99	Both	Multiple	Abadir	Good (ATPG)	Test vectors
ASPDAC'04	ED	Multiple	None	Good (SAT)	Test vectors
CoRé	**Both**	**Multi-ple**	**None**	**Good (SAT, simulation)**	**Test vectors**

Outline

- CoRé Framework
- Resynthesis techniques
  - Entropy-Guided Search (EGS)
  - Goal-Directed Search (GDS)
- Previous work
- Experimental results
- Conclusions

Experimental Results

- Enforcement of equivalency, 1024 initial vectors

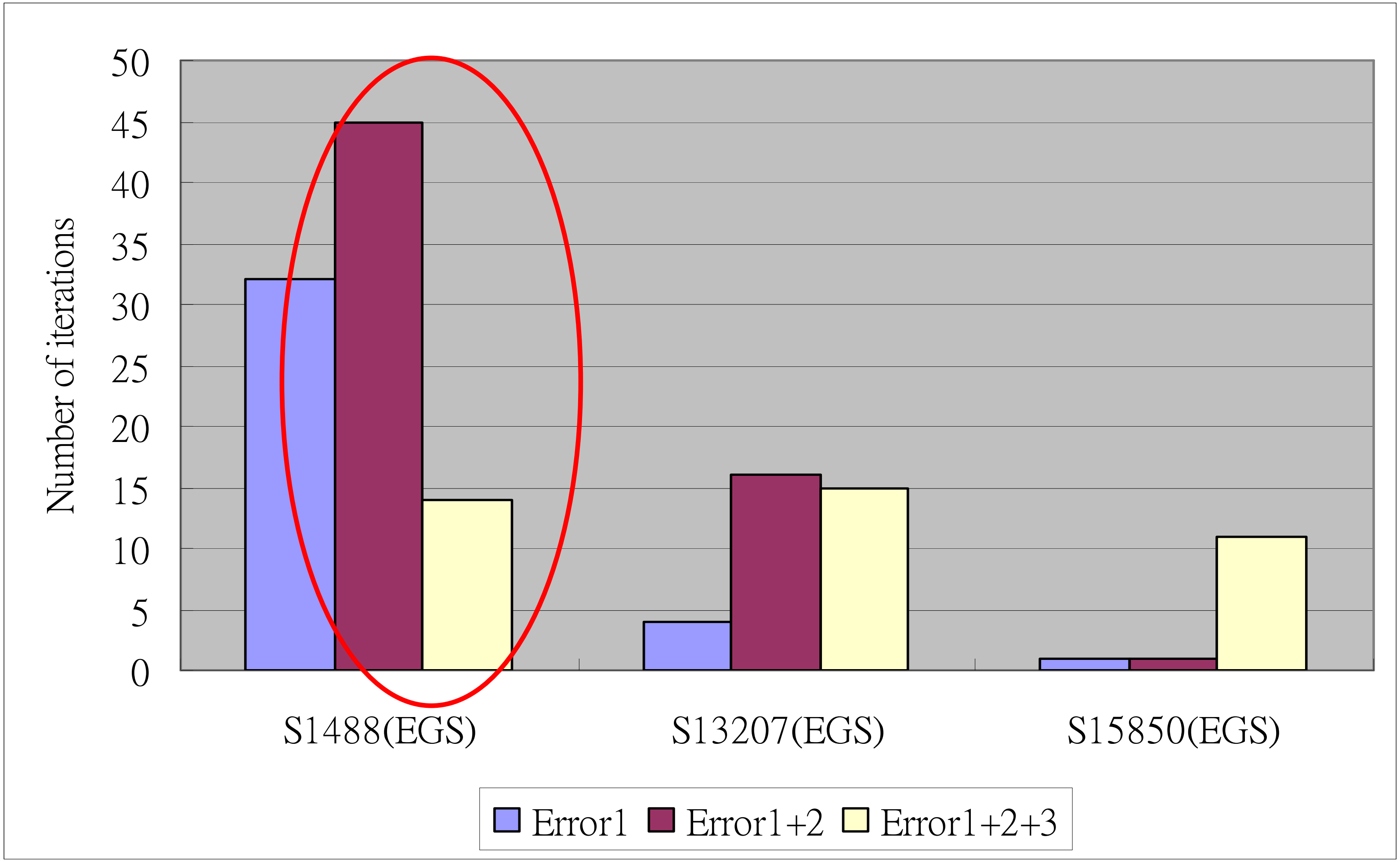
Bench-mark	Gate count	Type of error injected	EGS			
			Runtime (sec)			Number of iterations
			Error diagnosis	**Error correction**	Verifi-cation	
S1488	636	Gate change	4	1	1	1
S15850	685	Connection change	5	2	1	1
S13207	1219	Multiple gate change	6	1	1	1
S38584	6727	Gate change	306	1	81	1
AC97_Ctrl	11855	Multiple connection change	1032	2	252	5

Experimental Results

- Enforcement of equivalency
- Mimicking difficult errors with smaller number of initial vectors

Bench-mark	Initial vector number	EGS			
		Runtime (sec)			Number of iterations
		Error diagnosis	Error correction	Verifi-cation	
S1488	1024	4	1	1	1
S1488	64	4	1	1	3
S15850	1024	5	2	1	1
S15850	64	4	53	5	42
S9234_1	1024	9	1	1	1
S9234_1	64	10	1	3	4

Fixing Multiple Errors



Fixing Errors in Sequential Circuits

- Repair incorrect output responses of the given 32 bug traces
- Bugs were injected at the RTL

Benchmark	Description	#Cells	Bug description
Pre_norm	Part of FPU	1877	OR replaced by AND
MD5	MD5 full chip	13111	Incorrect state transition
DLX1	5-stage pipeline MIPS-Lite CPU	14725	JAL Inst. Leads to incorrect bypass from MEM stage
DLX2			Incorrect inst. forwarding

Benchmark	#Cycles	Err. Diag. time (sec)	EGS time (sec)
Pre_norm	20	136.3	2.7
MD5	10	5459	36.5
DLX1	47	69100	1703
DLX2	77	38261	77

Outline

- CoRé Framework
- Resynthesis techniques
  - Entropy-Guided Search (EGS)
  - Goal-Directed Search (GDS)
- Previous work
- Experimental results
- Conclusions

Conclusions

- CoRé framework
  - Based on abstraction and refinement of signatures
  - Only uses test vectors and output responses
    - Can be applied to most design flows
- An efficient simplification of SPFDs – Pairs of Bits to be Distinguished (PBDs)
  - Compactly encode information for resynthesis
- Effective resynthesis techniques
  - Entropy-Guided Search (EGS)
  - Goal-Directed Search (GDS)