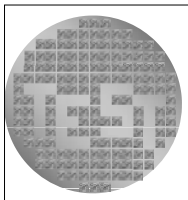


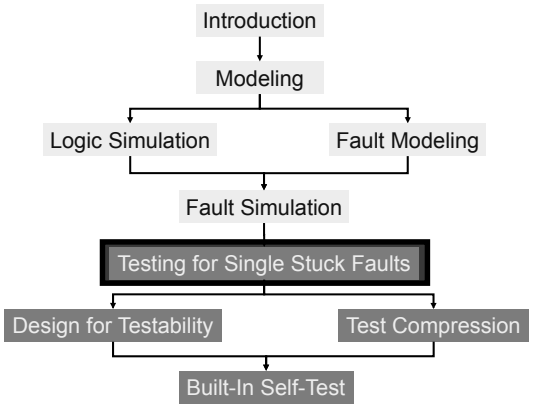
VLSI Test

Tsung-Chu Huang

Department of Electronic Engineering
National Changhua University of Education
Email: tch@cc.ncue.edu.tw

2007/11/02

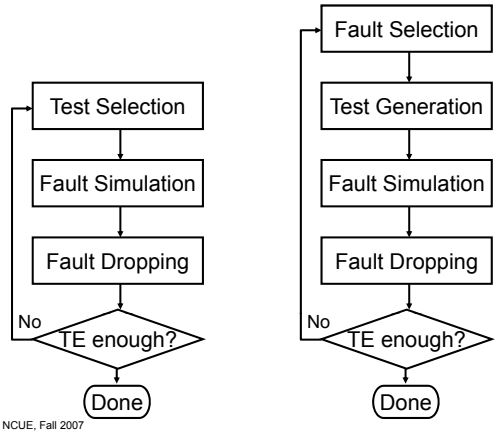
Syllabus & Chapter Precedence



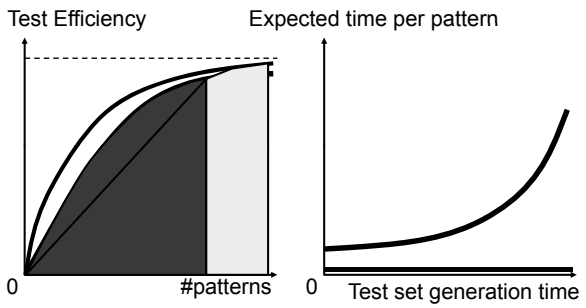
Testing for Single Stuck Faults

- Test Generation: Random vs. Diterministic
- ATPG for SSFs in Combinational Circuits
- ATPG for SSFs in Sequential Circuits

Test Generation: Random vs. Deterministic



Test Generation: Random vs. Deterministic



5-Value Operations

Notations

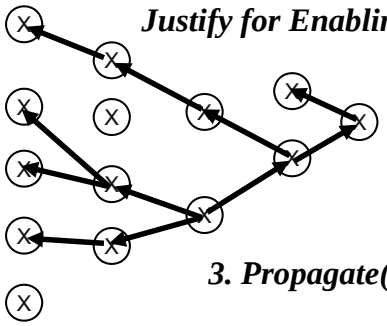
v/v_f		
0/0	0	0
1/1	1	0
1/0	D	↓
0/1	$\overline{D}$	↑

AND	0	1	D	$\overline{D}$	X
0	0	0	0	0	0
1	0	1	D	$\overline{D}$	X
D	0	D	D	0	X
$\overline{D}$	0	$\overline{D}$	0	$\overline{D}$	X
X	0	X	X	X	X

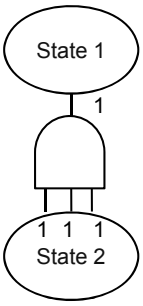
OR	0	1	D	$\overline{D}$	X
0	0	1	D	$\overline{D}$	X
1	1	1	1	1	1
D	D	1	D	1	X
$\overline{D}$	$\overline{D}$	1	1	$\overline{D}$	X
X	X	1	X	X	X

Test Generation for Fanout-Free Tree

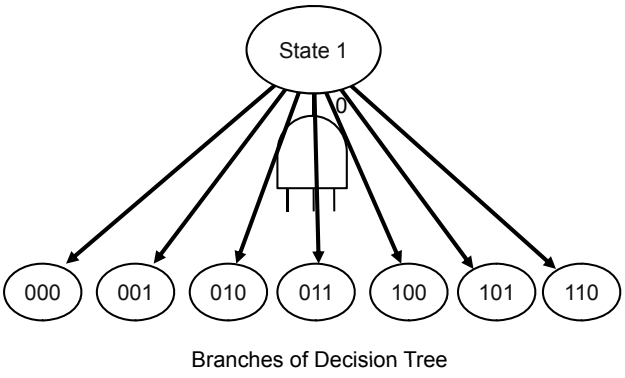
1. Set all values to X



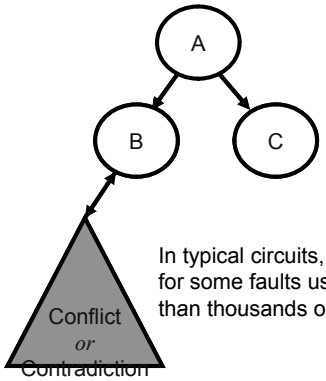
Decision Process in Justification



Decision Process in Justification



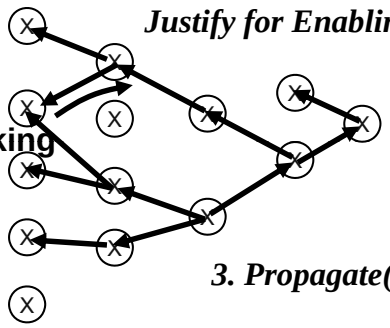
Backtracking in Decision Tree



In typical circuits, test generation for some faults usually have more than thousands of backtracking

Test Generation for Fanout-Free Tree

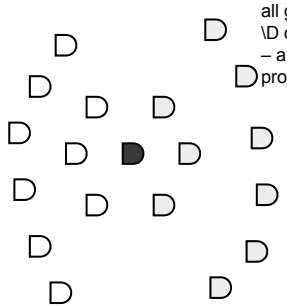
1. Set all values to X



Concept of Frontiers

J-frontier:
all gates keeping track of unsolved

D-frontier:
all gates with any D or \D on their inputs – a queue waiting for propagation.



J-Frontier **D-Frontier**

Basic Test Generation Algorithm

```
TG()
{
    if( imply_and_check( ) ) {
        if ( error_at_PO and all_lines_justified ) return(1);
        if ( no error can be propagated to a PO) return(0);
        select an unsolved problem;
        for (all ways) {
                select a untried way to solve it;
                if (TG( ) ) return (1);
        }
    }
    else return(0);
}
```


Implication Process

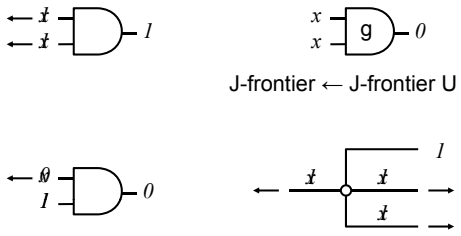
Purposes

1. Compute all values that can be uniquely determined by implication.
2. Check for consistency and assign values.
3. Maintain the D-frontier and the J-frontier.

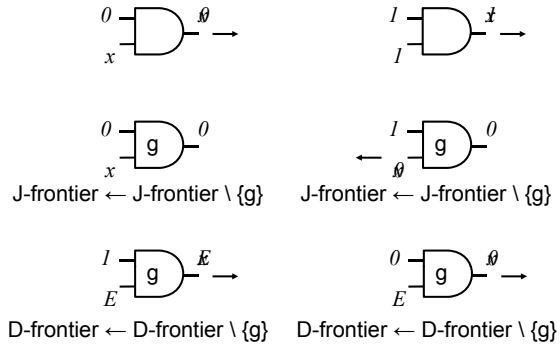
Operations

1. *Imply_and_check*() retrieves in tern every entry in the assignment queue (like the time wheel in logic simulation).

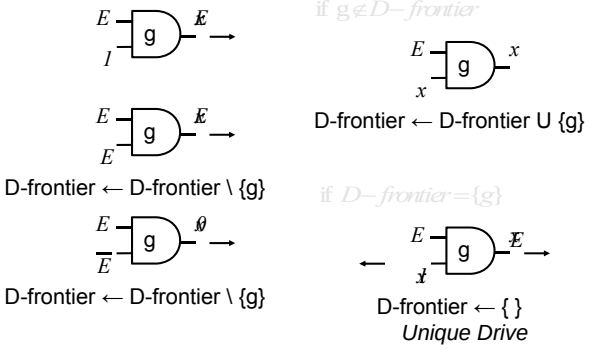
Backward Implication



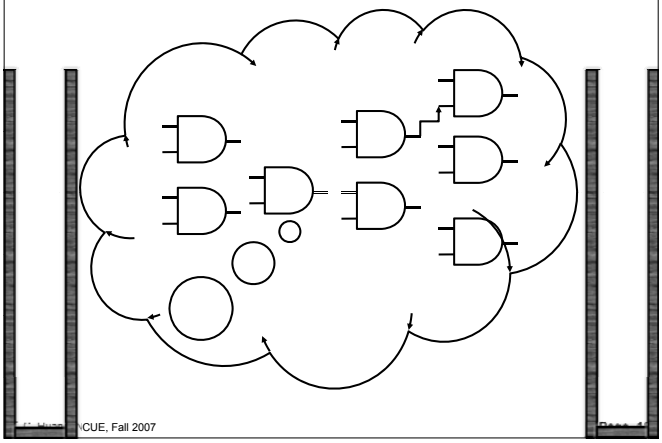
Forward Implication: Binary Values



Forward Implication: Error Values



Internal Model and Operating Space



D-Algorithm

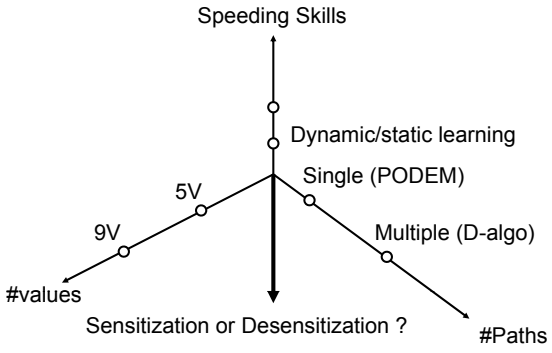
Roth 1966

```
D_Algo()
{
   if imply_and_check()
      if(error not at PO){
         repeat{
            select(g, D_frontier);
            enabling(g);
            if(D_algo())
return(SUCCESS);
         } until(all gates tried)

         return(FAILURE);
      }
   if(J_frontier)
      repeat{
         select(g, J_frontier);
         select(input_g);
         select(input_j, g);
         controlling(j, g);
         if(D_algo()) return(SUCCESS);
         enabling(j, g);
      } until(all gates tried);
   else return(SUCCESS);
   return(FAILURE);
}
```

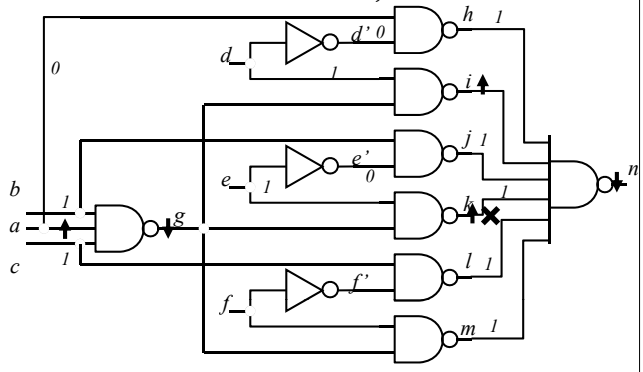
Characteristic feature:
Multiple-path sensitization, ability to propagate errors on several reconvergent paths.

Variety of PG Algorithms



Example for D-Algorithm

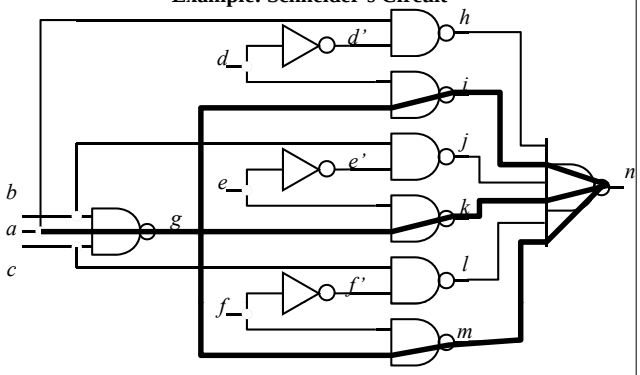
Schneider's Circuit, 1967



J-frontier:_____ D-frontier: g i k m n

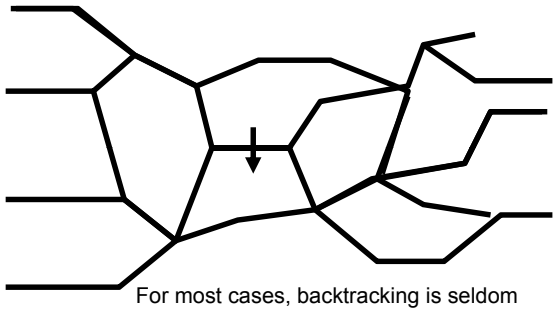
Multiple-Path Sensitization

Example: Schneider's Circuit



Multiple-Path Sensitization

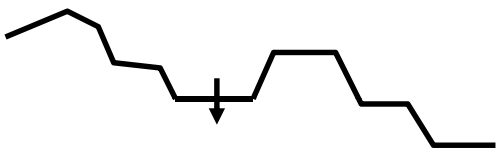
Conceptual Graph



For most cases, backtracking is seldom
And even single path is sufficient.

Single-Path Sensitization

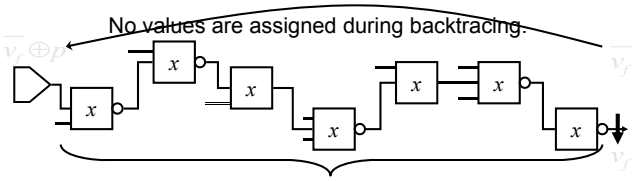
Conceptual Graph



Breadth-First Searching → Depth First Searching
Consider only further error propagation
and ignore the other gates from D-frontier.

Backtracing

Direct Search/Mapping objective to PI assignment



Inversion parity p: 1 if the inversion number is odd, otherwise 0.

```
Backtrace(k, v_k)
{
        v←v_k;
        while(k not PI){
                select an input j of k with value x;
                v← v ⊕inversion(k);
                k← j;}
        return(k, v);
}
```


PODEM

Path-Oriented Decision Making [Goel 1981]

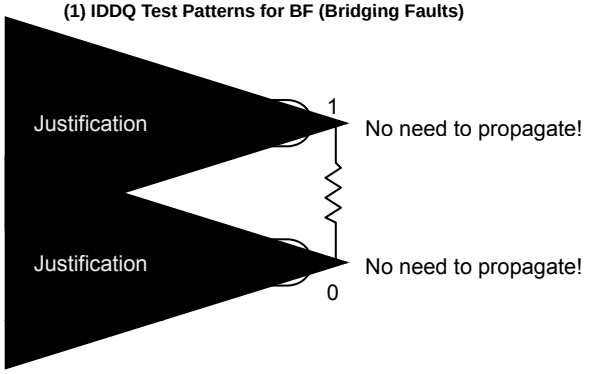
```
PODEM( )
{
        if(Error at PO) return(Success);
        if(test not possible) return(Failure);
        select an objective (k, v_k) from D-frontier;
        (j, v_j) = Backtrace(k, v_k);
        Imply (j, v_j);
        if(PODEM( )) return(Success);
        Imply(j, not v_j);
        if(PODEM( )) return(Success);
        Imply(j, x);
        return Failure;
}
```

No Need to

- **Consistency Check**
- **J-Frontier**
- **Backward Implication**

Implantation of STF ATPG

(1) IDDQ Test Patterns for BF (Bridging Faults)



Implantation of STF ATPG

(2) Transition Delay Faults (eg., Slow-to-Rise)