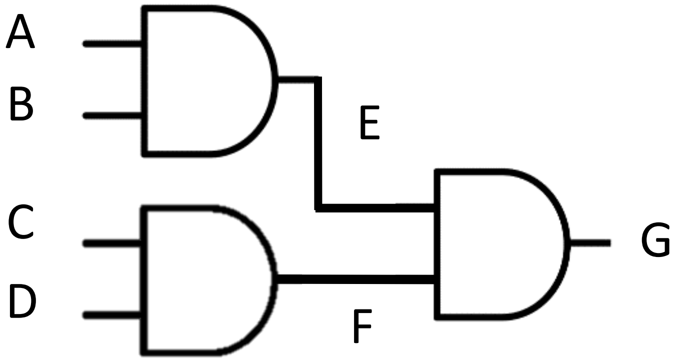
Introduction to Path Delay Fault And A Sat-Based Path-Delay-Fault ATPG Solver

VLSI Testing LAB: ED612
陳玥融 rosechen30932@gmail.com
陳柏諺 chentt13@gmail.com

Path Delay Fault

- □ Any path with a total delay exceeding the clock interval is called a "path fault."
- □ One major problem in finding delay faults is the number of possible paths in a circuit under test (CUT). The number of total paths can grow exponentially with circuit size.



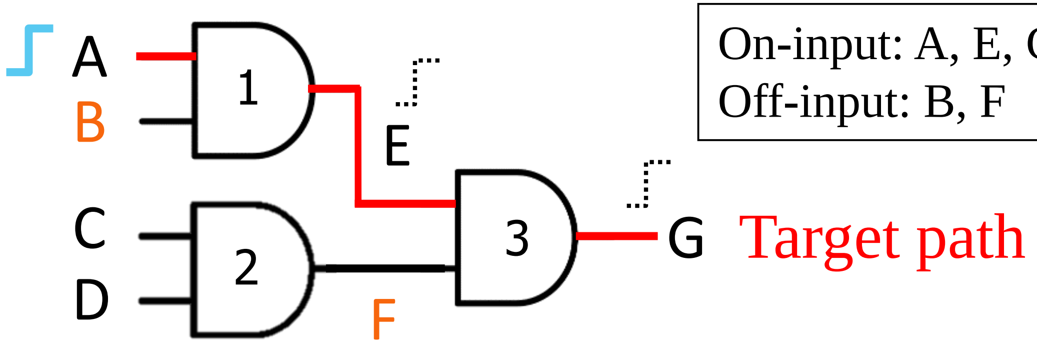
4 paths in the circuit:

- A → E → G
- B → E → G
- C → F → G
- D → F → G

Path-Delay-Fault ATPG

- A path-delay-fault ATPG generates a vector pair (v1,v2) that can distinguish between the correct circuit behavior and the faulty circuit behavior caused by defects.
- 2 phases of ATPG:
  - Fault Activation: Creates a transition (either rising or falling) at the beginning of the target path.
  - Fault Propagation: Propagates the corresponding transition to primary output through the target path.

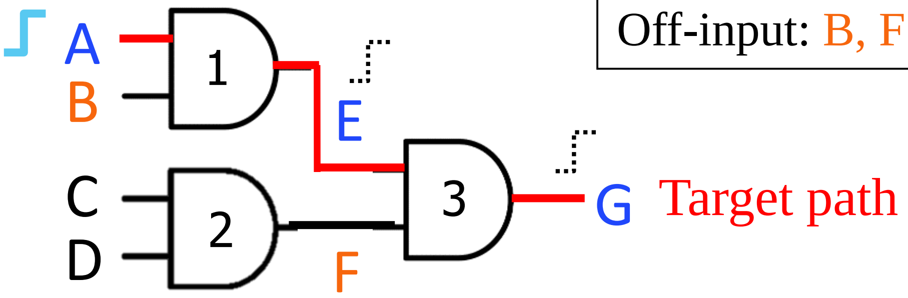
Output of ATPG:
(v1, v2)
Ex. 0111，1111



On-input: A, E, G
Off-input: B, F

Some Terms in Path-Delay-Fault ATPG

- On-input & off-input



On-input: A, E, G
Off-input: B, F

- Controlling Values & Non-Controlling Values

Gate Type	Controlling Value (CV)	Non-Controlling Value (NCV)
AND	0	1
OR	1	0

- Timeframe

Transition Type	Logic value at Timeframe 0	Logic value at Timeframe 1
Rising(0→1)	0	1
Falling(1→0)	1	0

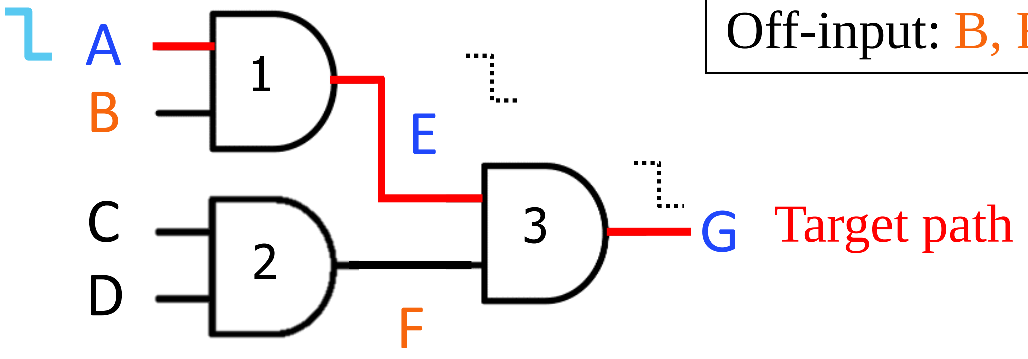
Test Quality for A Path

- □ A test pattern (v1,v2) of a path P can be classified as Robust or Non-Robust based on the transition states on the on-inputs and off-inputs.

On-input: A, E, G
Off-input: B, F

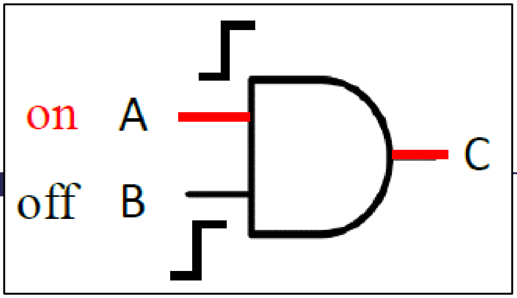
Output of ATPG:
(v1, v2)

R: 1111，0111
NR: 1XXX，0111

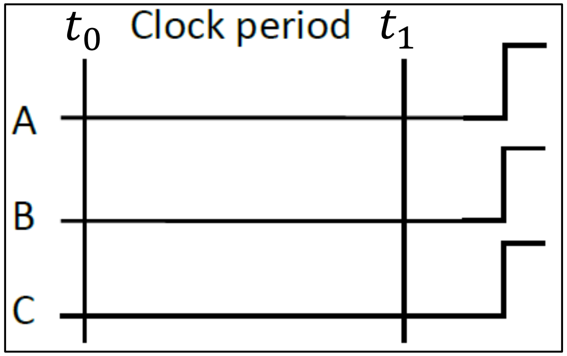


On-input transition	Off-input assignments	
	Robust(R)	Non-Robust(NR)
cv → ncv	x → ncv	x → ncv
ncv → cv	ncv → ncv	x → ncv

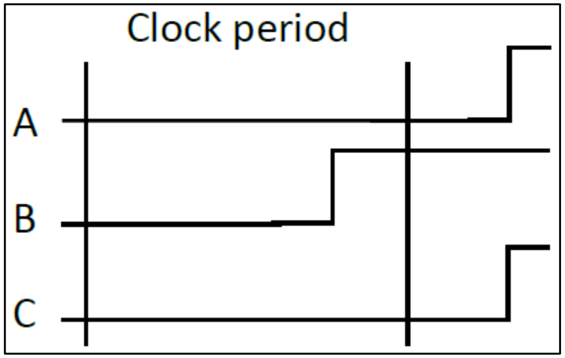
Robust Test



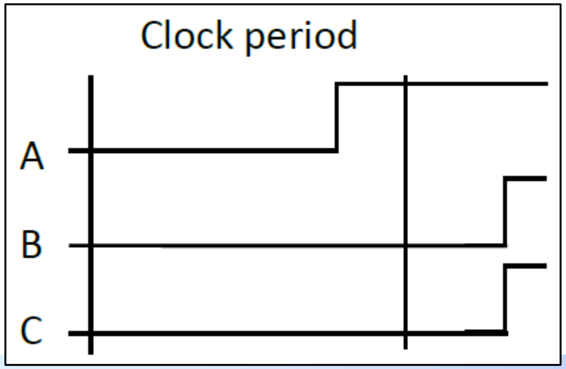
	t_0	t_1
A	0	1
B	0	1
C	0	1



- □ Both inputs have delay fault
  - Success

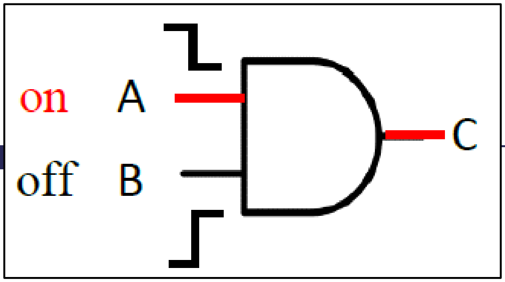


- □ Only on-input has delay fault
  - Success

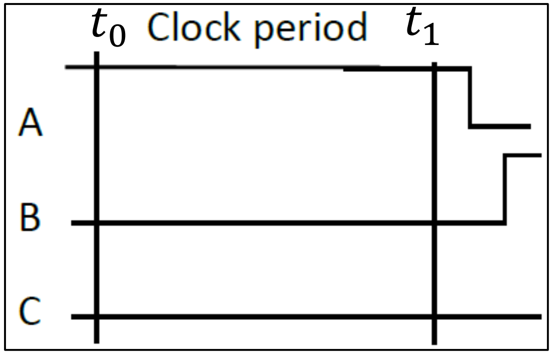


- □ Only off-input has delay fault
  - Success

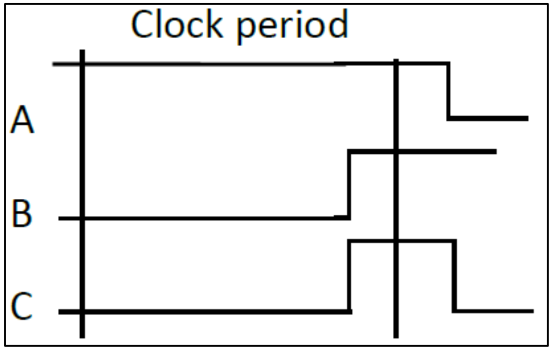
Non-Robust Test



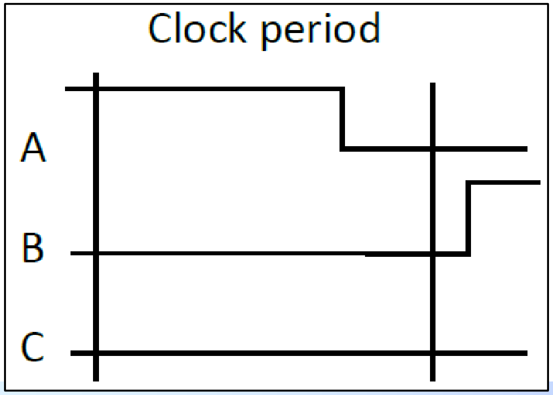
	t_0	t_1
A	1	0
B	0	1
C	0	0



- □ Both inputs have delay fault
  - ■ Fail



- □ Only on-input has delay fault
  - ■ Success



- □ Only off-input has delay fault
  - ■ Fail

Boolean Satisfiability (SAT) Problem

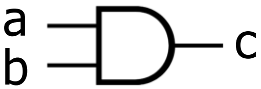
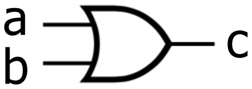
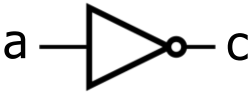
- SAT is the problem of determining if there exists an interpretation that satisfies a given Boolean formula.
  - (¬ a ∧ b) is satisfiable because a=0 and b=1 make it true.
  - (¬ a ∧ a) is unsatisfiable.
- SAT is the first problem that was proven to be NP-complete which means there is no known algorithm that efficiently solves each SAT problem.
- A 3-CNF-SAT problem: Given a Boolean formula $\mathbb{F}$ in 3-CNF form, does there exist any interpretation that make $\mathbb{F}$ true?

Conjunctive Normal Form (CNF)

- A literal is either a Boolean variable or the negation of a Boolean variable.
  - $x, \neg y$
- A clause is a disjunction of literals.
  - $(x \vee y \vee \neg z)$
- A Boolean formula is in Conjunctive Normal Form if it is a conjunction of clauses (or a single clause).
  - $(x \vee y \vee \neg z \vee w \vee \neg u) \wedge (\neg y \vee z) \wedge (\neg x)$
  - $(x \vee y \vee \neg z \vee w \vee \neg u)$
- 3-CNF: a conjunction of clauses where each clause contains exactly 3 literals/at most 3 literals.
  - $(x \vee y \vee \neg z) \wedge (\neg y \vee z) \wedge (\neg x)$

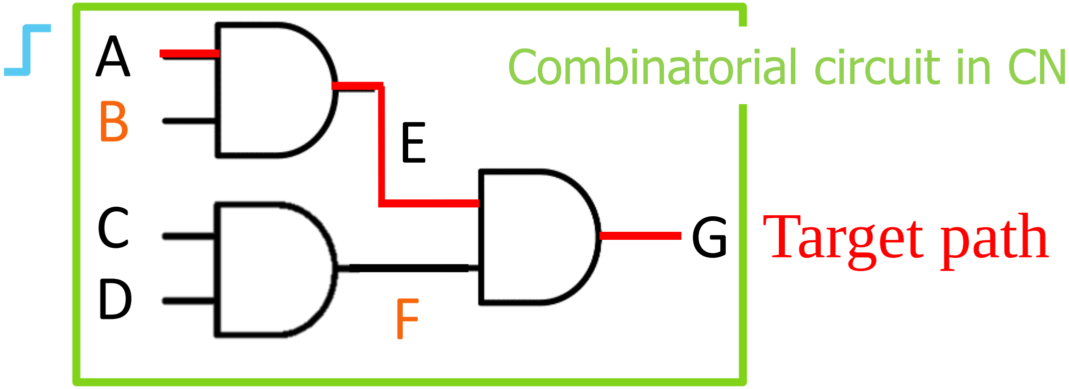
Tseytin Transformation

- Tseytin transformation takes as input an arbitrary combinatorial logic circuit and produces a Boolean formula in conjunctive normal form (CNF), which can be solved by a CNF-SAT solver.

Gate type	**Operation**	**CNF expressions**
	c = a · b	(¬a ∨ ¬b ∨ c) ∧ (a ∨ ¬c) ∧ (b ∨ ¬c)
	c = a + b	(a ∨ b ∨ ¬c) ∧ (¬a ∨ c) ∧ (¬b ∨ c)
	c = -a	(¬a ∨ ¬c) ∧ (a ∨ c)

SAT-Based Path-Delay-Fault ATPG

- A given combination circuit + a given PDF.
- Do Fault Activation & Fault Propagation = Build a Boolean formula in CNF and feed into the SAT solver.
  - Problem definition: Can we find a test vector pair to detect the transition delay fault rising on path A → E → G ?



For non-robust test:

1st timeframe: (¬A ∨ ¬B ∨ E) ∧ (A ∨ ¬E) ∧ (B ∨ ¬E) ∧ ….. ∧ ¬A

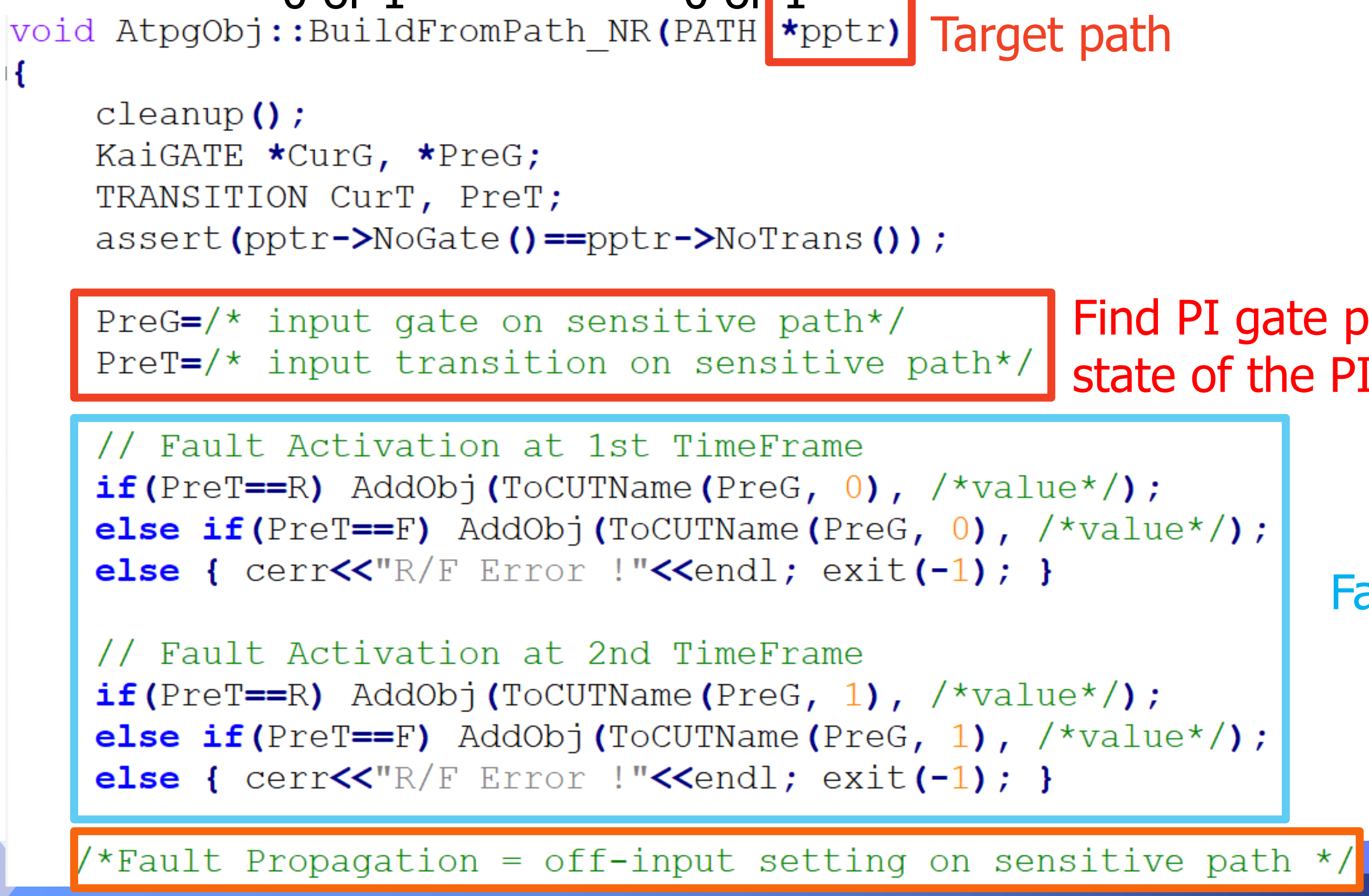
2nd timeframe: (¬A ∨ ¬B ∨ E) ∧ (A ∨ ¬E) ∧ (B ∨ ¬E) ∧ ….. ∧ A ∧ B ∧ F

- If the SAT problem is solved, a test vector pair(v1, v2) is successively found and the PDF is detected.

What you need to do in HW4 In kai_objective.cpp

- Add more clauses to the original combinational circuit CNF by using AddObj(ToCUTName(gate ptr, timeframe), logic value)

0 or 1 0 or 1



```
void AtpgObj::BuildFromPath_NR(PATH *pptr)
{
    cleanup();
    KaiGATE *CurG, *PreG;
    TRANSITION CurT, PreT;
    assert(pptr->NoGate()==pptr->NoTrans());

    PreG=/* input gate on sensitive path*/
    PreT=/* input transition on sensitive path*/

    // Fault Activation at 1st TimeFrame
    if(PreT==R) AddObj(ToCUTName(PreG, 0), /*value*/);
    else if(PreT==F) AddObj(ToCUTName(PreG, 0), /*value*/);
    else { cerr<<"R/F Error !"<<endl; exit(-1); }

    // Fault Activation at 2nd TimeFrame
    if(PreT==R) AddObj(ToCUTName(PreG, 1), /*value*/);
    else if(PreT==F) AddObj(ToCUTName(PreG, 1), /*value*/);
    else { cerr<<"R/F Error !"<<endl; exit(-1); }

/*Fault Propagation = off-input setting on sensitive path */
```

Target path

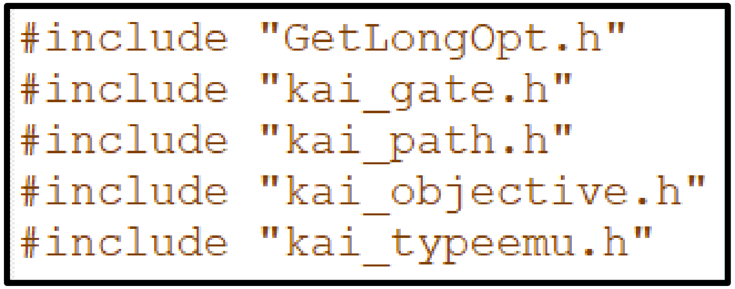
Find PI gate ptr and the transition state of the PI on target path

Fault Activation

Fault Propagation

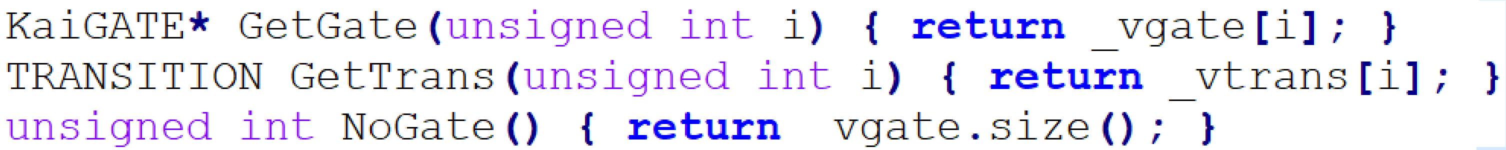
More Hints

- □ Go to the following files to understand the data structure of this program.



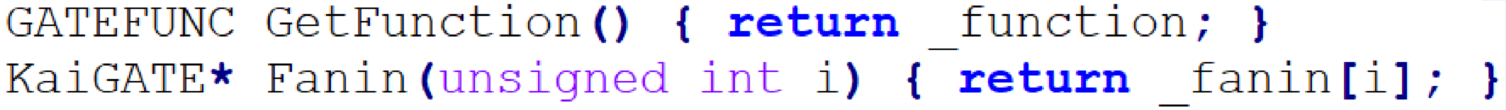
```
#include "GetLongOpt.h"
#include "kai_gate.h"
#include "kai_path.h"
#include "kai_objective.h"
#include "kai_typeemu.h"
```

  - ■ E.g. In kai_path.h:



```
KaiGATE* GetGate(unsigned int i) { return _vgate[i]; }
TRANSITION GetTrans(unsigned int i) { return _vtrans[i]; }
unsigned int NoGate() { return  vgate.size(); }
```

  - ■ E.g. In kai_gate.h:



```
GATEFUNC GetFunction() { return _function; }
KaiGATE* Fanin(unsigned int i) { return _fanin[i]; }
```

- □ Only AND, NAND, OR, NOR gates need to be considered.