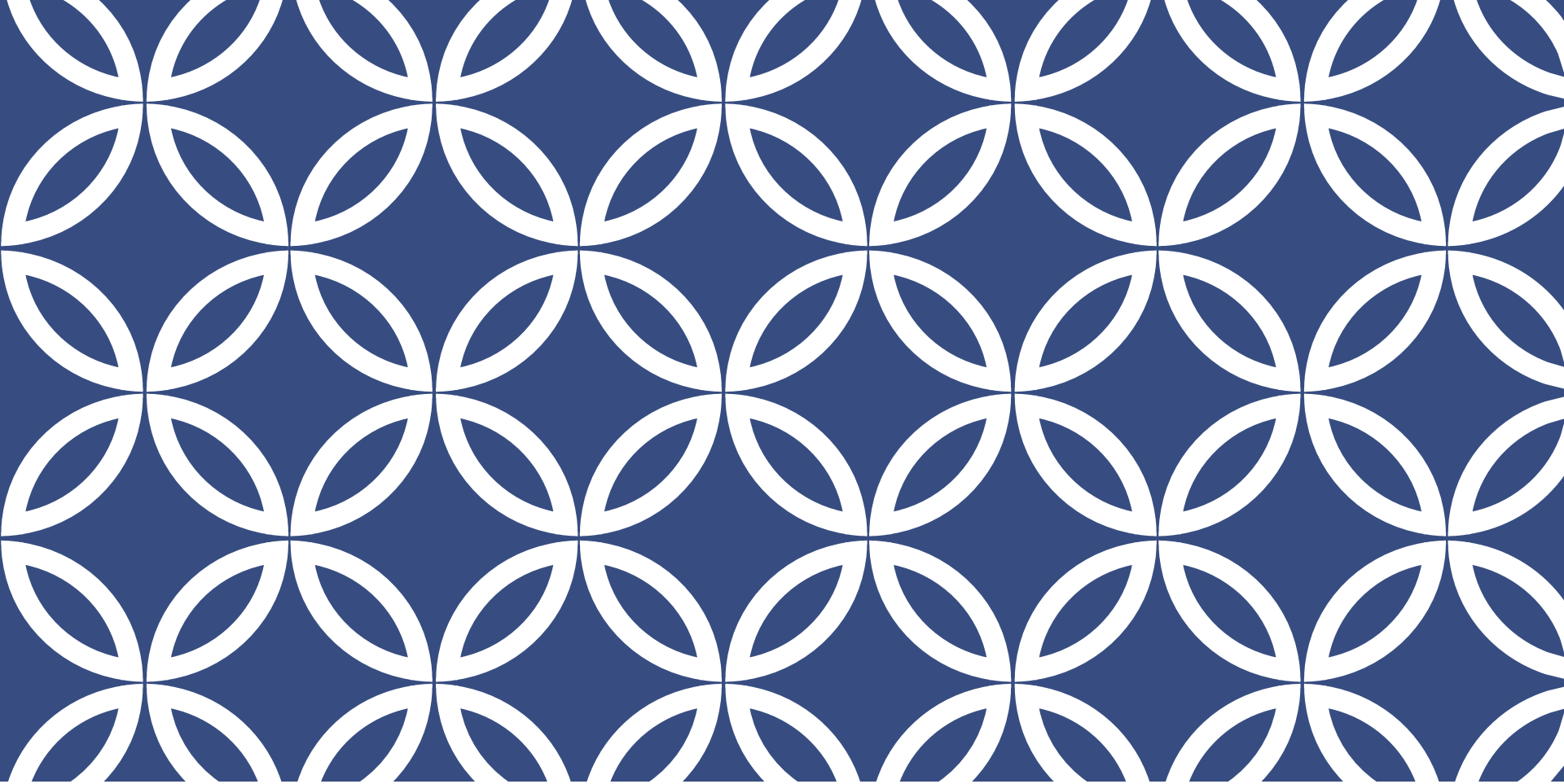


IEE5650 VLSI Testing PODEM Tutorial

Yu-Teng Nien

Introduction

- PODEM
  - An automatic test pattern generation (ATPG) tool for stuck-at/transition fault models
  - Support logic simulation and fault simulation functions
- We use PODEM in this course
  - To implement various testing algorithms in assignments
  - Basically add functions to the unfinished codebase
- Requirements
  - Basic data structure concepts
    - Vector, linked list, map, hash
  - C++ programming ability
    - Class, data member, member function

Environment

- Under Linux-like system
  - GCC 3.x
  - make 3.8+
  - flex 1.875+
  - bison 2.5.4+
  - Library
    - Readline 5.0.4+
    - Ncurses 5.4.2+

Setup (1/2)

- Download source code from *Assignment #0* at course website
  - podem.tgz
  - circuits.tgz
- Decompress the tarball

  $ tar zxvf podem.tgz

  $ tar zxvf circuits.tgz
- Check file list

  $ ls podem/

Setup (2/2)

- Change directory

  $ cd podem

- Compile the source code with *Makefile*

  $ make

- An executable file *atpg* will be generated

  $ ls atpg

- Invocation

  $ ./atpg

How to Use PODEM

- Show usage

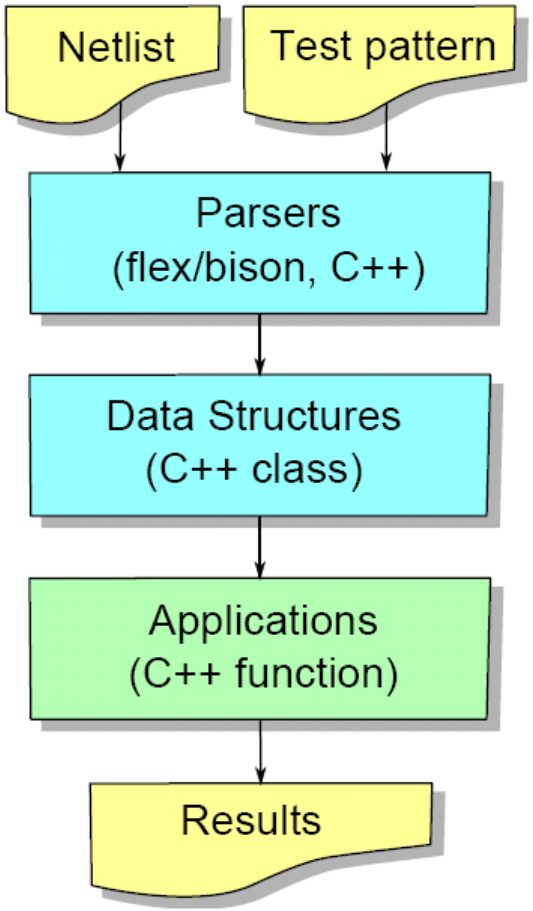
  $ ./atpg -help

  usage: atpg [options] input_circuit_file

  -help (print this help summary)
  -logicsim (run logic simulation)
  -plogicsim (run parallel logic simulation)
  -fsim (run stuck-at fault simulation)
  -stfsim (run single pattern single transition-fault simulation)
  -transition (run transition-fault ATPG)
  -input <$val> (set the input pattern file)
  -output <$val> (set the output pattern file)
  -bt [$val] (set the backtrack limit)

- Refer to *README.pdf* under podem/ for more function description and examples

PODEM Scheme



- Input files
  - Netlist and Test pattern
- Parser
  - To translate files as data structures
- Data structures
  - CIRCUIT, GATE, and etc
- **Applications**
  - Logic/fault simulator and ATPG
  - Do homeworks in this layer
- Results
  - Fault coverage and test patterns
  - Performance information

Arguments (1/2)

- Command line interface example
  - Logic simulation
    - ./atpg -logicsim -input <pattern> <circuit>
- Setup (the SetupOption function in *main.cc*)
  - option.enroll(Name, InputValue, Description, 0);
  - InputValue
    - GetLongOpt::NoValue
    - GetLongOpt::MandatoryValue
    - GetLongOpt::OptionalValue
- Example
  - option.enroll("logicsim", GetLongOpt::NoValue,
    "run logic simulation", 0);
  - option.enroll("input", GetLongOpt::MandatoryValue,
    "set the input pattern file", 0);

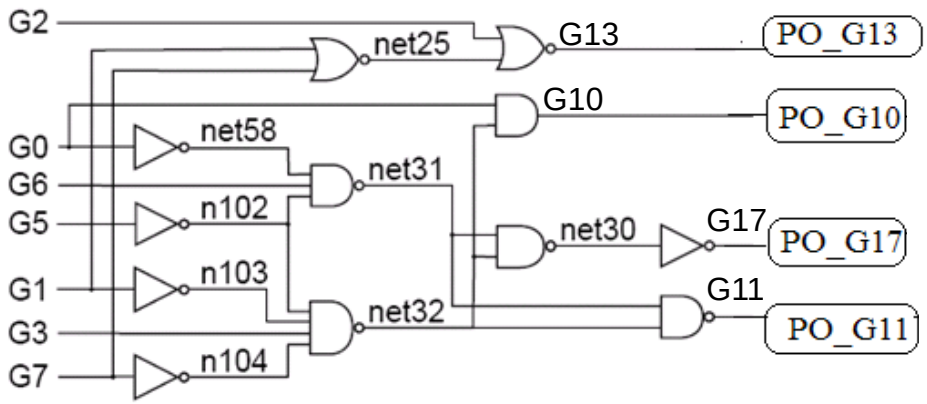
Arguments (2/2)

- Usage (in *main.cc*)
  - Check if certain option is specified in command line
  - option.retrieve(name_of_agrument)
- Example

```
// if option “-logicism” is set
if (option.retrieve("logicsim")) {
    // code for logic simulator
}
```

ISCAS89 Netlist Format

- Supported gate types
  - INPUT, OUTPUT, AND, NAND, OR, NOR, NOT, DFF
- s27.bench schematic




```
# s27.bench
INPUT(G0)
INPUT(G1)
INPUT(G2)
INPUT(G3)
<skip>
OUTPUT(G17)
OUTPUT(G10)
OUTPUT(G11)
<skip>
G17 = NOT(net30)
G10 = AND(net32, G0)
G11 = NAND(net32, net31)
<skip>
net25 = NOR(G7, G1)
net30 = NAND(net31, net32)
```

Pattern Format

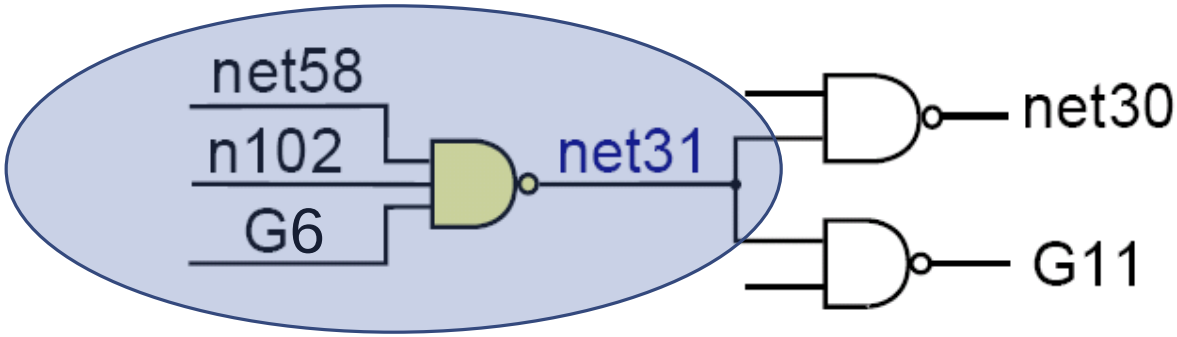
- Primary input (PI) names in the first line
- Primary input values in the others
- Example: 7 test patterns for s27.bench
  - PI G0 PI G1 PI G2 PI G3 PI G5 PI G6 PI G7
  - 1100011
  - 0110010
  - 1001011
  - 1001010
  - 1101010
  - 0011111
  - 1011100

Data Structures – Class GATE

- Data members used in assignment #0 (in *gate.h*)
  - Name: gate name
  - Function:
    G_PI, G_PO, G_PPI, G_PPO,
    G_NOT, G_BUF, G_DFF,
    G_AND, G_NAND, G_OR, G_NOR
  - Input_list/Output_list:
    fanin/fanout gate list
- Data members used in later assignments
  - Value, Flag, Fault status, ATPG, status, etc

Data Structures – GATE Example

- Gate is named after its output net
- Example: Gate net31
  - Name = net31
  - Function = G_NAND
  - Input_list = {net58, n102, G6}
  - Output_list = {net30, G11}
- Net counts
  - #Stem nets: 1
    - net31
  - #Branch nets: 2
    - net31 -> net30
    - net31 -> G11



`net31 = NAND(net58, n102, G6)` in s27.bench

Benchmark Circuits

- circuits/
  - iscas85/
    - cXXX.bench is the combinational ISCAS85 circuits
  - iscas89_seq/
    - sXXX_seq.bench is the original ISCAS89 circuits
  - iscas89_opt/
    - sXXX_opt.bench is the combinational parts of sXXX_seq.bench optimized with the Synopsys DC
  - iscas89_com/
    - sXXX_com.bench is the same as sXXX_opt.bench (We do not need these circuits at all.)