積體電路系統測試 期末Project

作者:
B0990198 吳佳政 B09901153 胡凱翔 R12943163 邱楷翔

Problem 1 Description

1. A list of fully specified test pattern and circuit netlist information for 3 circuits has been given
2. Task: Find the least amount of test patterns that would achieve test coverage of 60-90% for all circuits

Solution

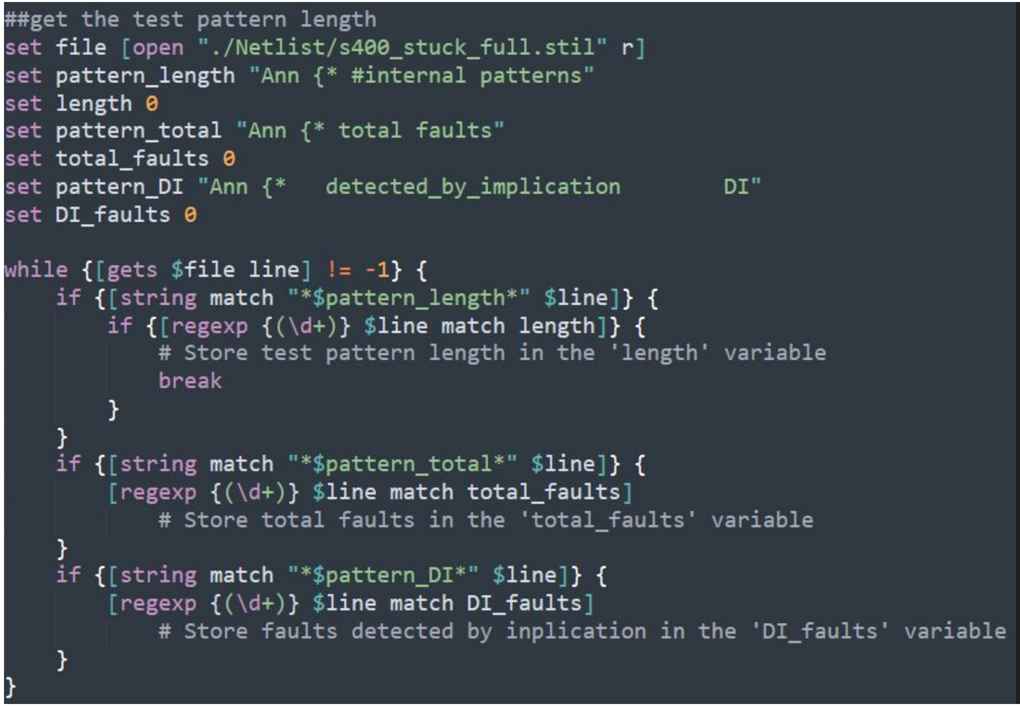
Core algorithm:

1. For each fault assign it to its respective detectable patterns
2. Select pattern with most faults detected, for every fault of said pattern eliminate this fault from all other patterns that has this fault and add faults from this pattern to test coverage
3. Repeat step 2 until fault coverage demand has been met
4. O(PFP) time (P is the number of patterns, F is the number of faults)

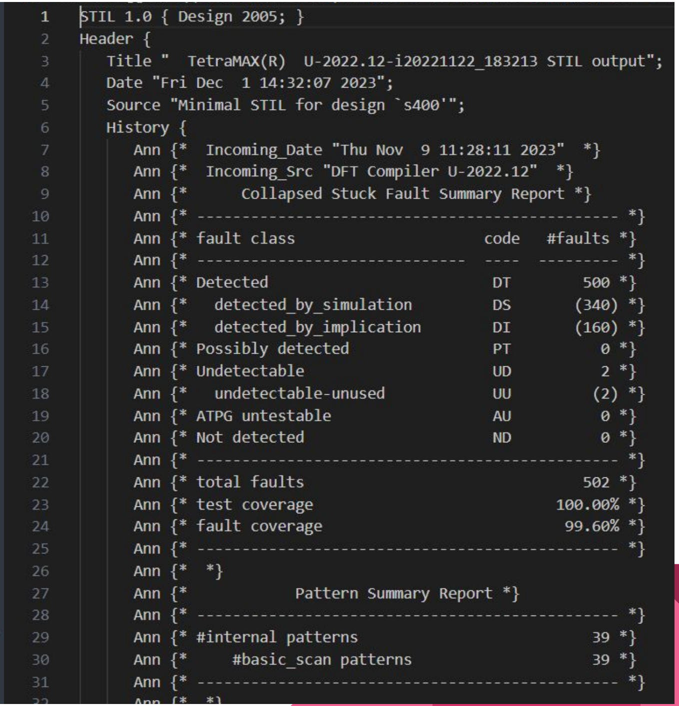
Technical Details

1. TCL file
2. C++ file
    a. main.cpp
    b. makefile
    c. tm_usage.cpp

TCL file

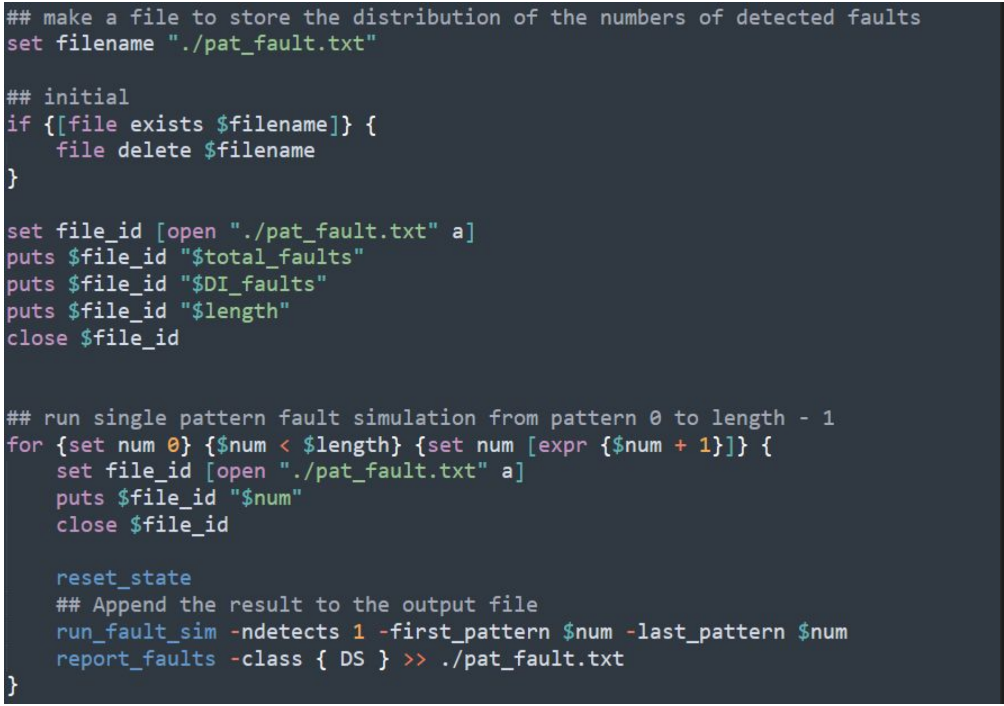


```
##get the test pattern length
set file [open "./Netlist/s400_stuck_full.stil" r]
set pattern_length "Ann {* #internal patterns"
set length 0
set pattern_total "Ann {* total faults"
set total_faults 0
set pattern_DI "Ann {*   detected_by_implication        DI"
set DI_faults 0

while {[gets $file line] != -1} {
    if {[string match "*$pattern_length*" $line]} {
        if {[regexp {(\d+)} $line match length]} {
            # Store test pattern length in the 'length' variable
            break
        }
    }
    if {[string match "*$pattern_total*" $line]} {
        [regexp {(\d+)} $line match total_faults]
            # Store total faults in the 'total_faults' variable
    }
    if {[string match "*$pattern_DI*" $line]} {
        [regexp {(\d+)} $line match DI_faults]
            # Store faults detected by inplication in the 'DI_faults' variable
    }
}
```

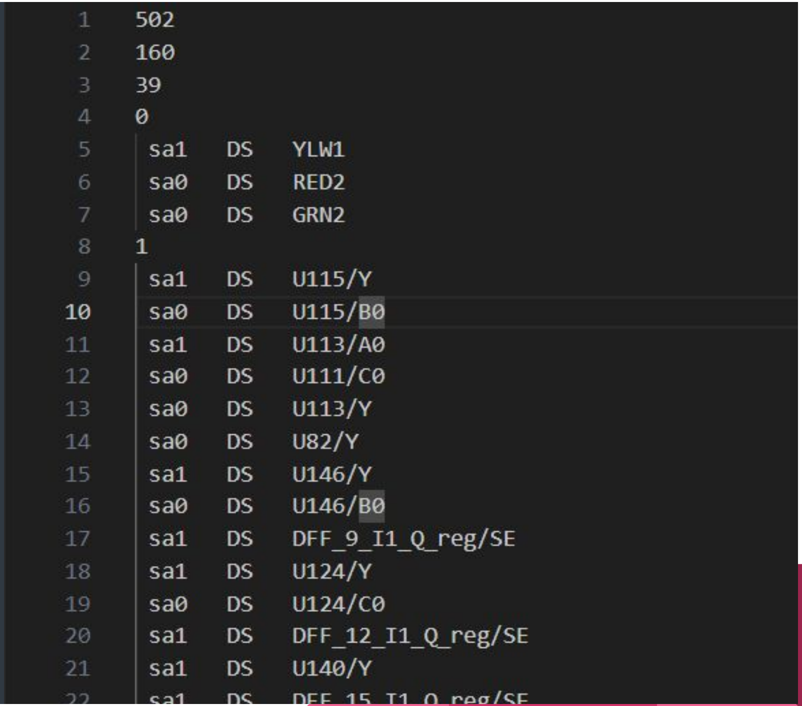


```
STIL 1.0 { Design 2005; }
Header {
   Title "  TetraMAX(R)  U-2022.12-i20221122_183213 STIL output";
   Date "Fri Dec  1 14:32:07 2023";
   Source "Minimal STIL for design `s400'";
   History {
      Ann {*  Incoming_Date "Thu Nov  9 11:28:11 2023"  *}
      Ann {*  Incoming_Src "DFT Compiler U-2022.12"  *}
      Ann {*      Collapsed Stuck Fault Summary Report *}
      Ann {* ----------------------------------------------- *}
      Ann {* fault class                     code   #faults *}
      Ann {* ------------------------------  ----  --------- *}
      Ann {* Detected                         DT         500 *}
      Ann {*   detected_by_simulation         DS       (340) *}
      Ann {*   detected_by_implication        DI       (160) *}
      Ann {* Possibly detected                PT           0 *}
      Ann {* Undetectable                     UD           2 *}
      Ann {*   undetectable-unused            UU         (2) *}
      Ann {* ATPG untestable                  AU           0 *}
      Ann {* Not detected                     ND           0 *}
      Ann {* ----------------------------------------------- *}
      Ann {* total faults                                502 *}
      Ann {* test coverage                           100.00% *}
      Ann {* fault coverage                           99.60% *}
      Ann {* ----------------------------------------------- *}
      Ann {*  *}
      Ann {*             Pattern Summary Report *}
      Ann {* ----------------------------------------------- *}
      Ann {* #internal patterns                           39 *}
      Ann {*     #basic_scan patterns                     39 *}
      Ann {* ----------------------------------------------- *}
      Ann {*  *}
```

TCL file

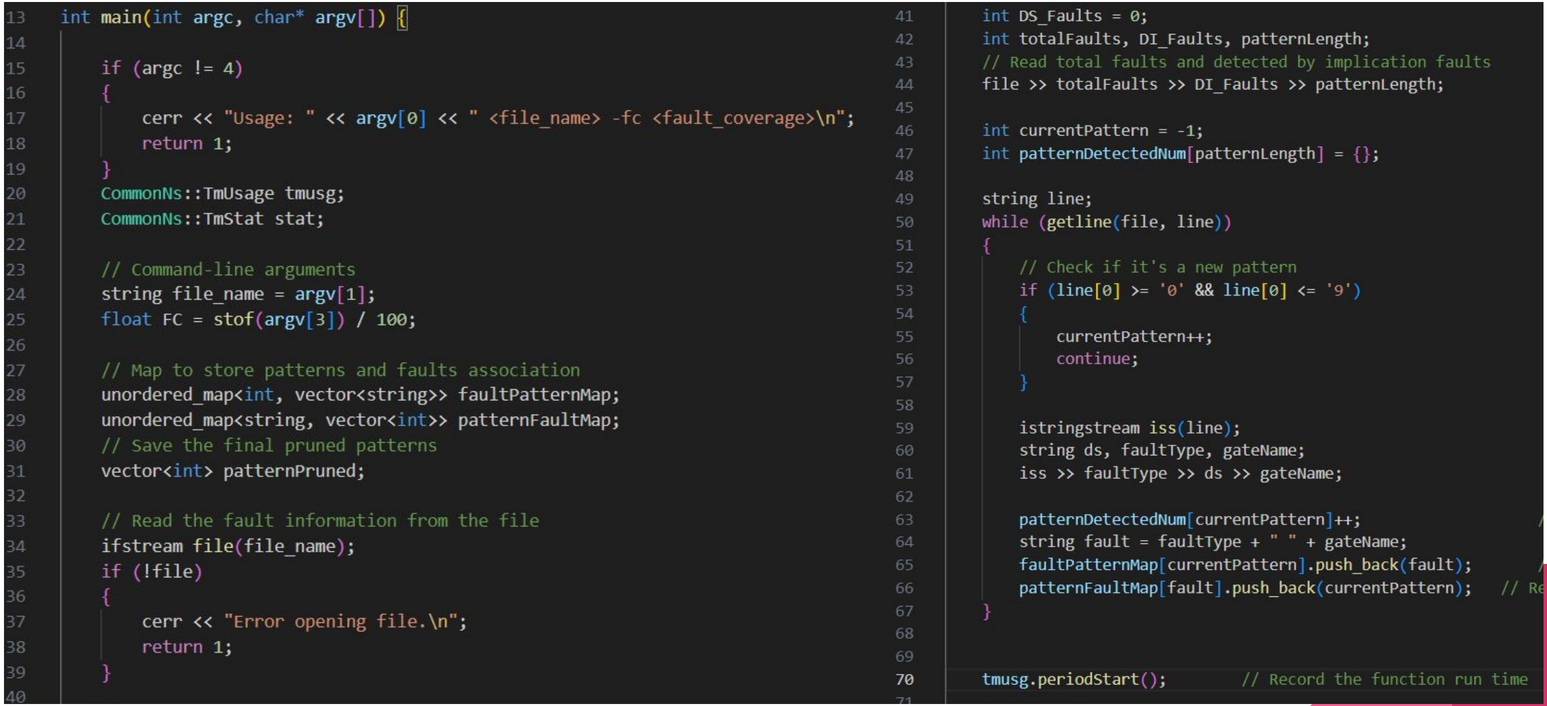


```
## make a file to store the distribution of the numbers of detected faults
set filename "./pat_fault.txt"

## initial
if {[file exists $filename]} {
    file delete $filename
}

set file_id [open "./pat_fault.txt" a]
puts $file_id "$total_faults"
puts $file_id "$DI_faults"
puts $file_id "$length"
close $file_id


## run single pattern fault simulation from pattern 0 to length - 1
for {set num 0} {$num < $length} {set num [expr {$num + 1}]} {
    set file_id [open "./pat_fault.txt" a]
    puts $file_id "$num"
    close $file_id

    reset_state
    ## Append the result to the output file
    run_fault_sim -ndetects 1 -first_pattern $num -last_pattern $num
    report_faults -class { DS } >> ./pat_fault.txt
}
```



```
502
160
39
0
  sa1   DS   YLW1
  sa0   DS   RED2
  sa0   DS   GRN2
1
  sa1   DS   U115/Y
  sa0   DS   U115/B0
  sa1   DS   U113/A0
  sa0   DS   U111/C0
  sa0   DS   U113/Y
  sa0   DS   U82/Y
  sa1   DS   U146/Y
  sa0   DS   U146/B0
  sa1   DS   DFF_9_I1_Q_reg/SE
  sa1   DS   U124/Y
  sa0   DS   U124/C0
  sa1   DS   DFF_12_I1_Q_reg/SE
  sa1   DS   U140/Y
  sa1   DS   DFF_15_I1_Q_reg/SE
```

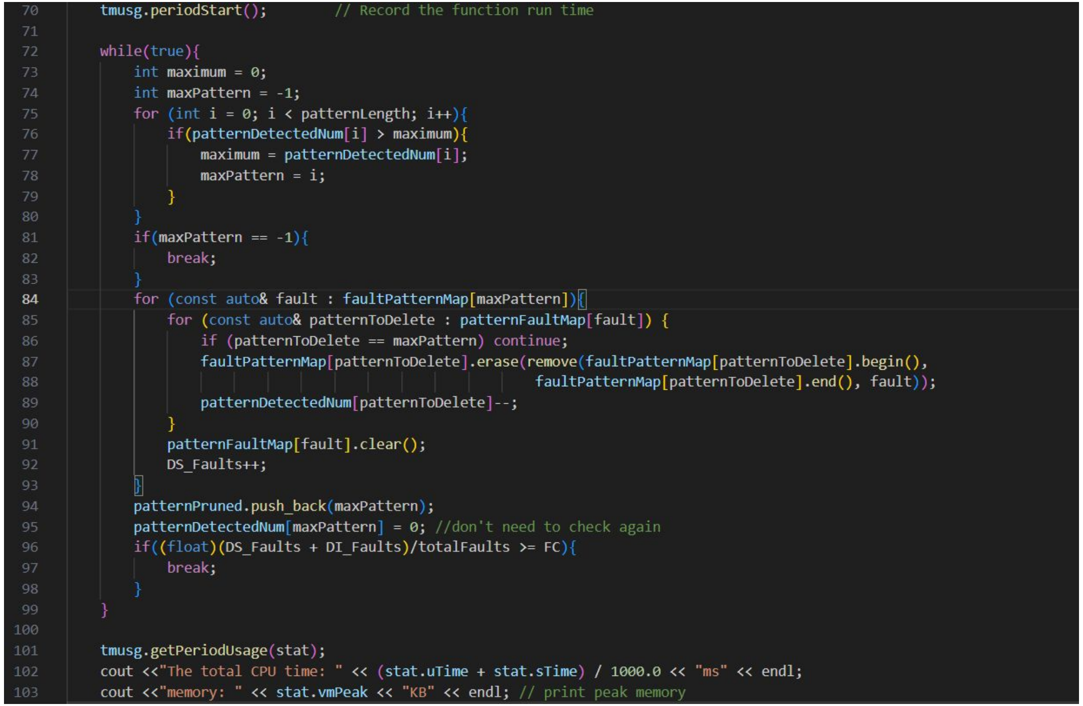
Get input file(generated by .tcl file)



```cpp
int main(int argc, char* argv[]) {

    if (argc != 4)
    {
        cerr << "Usage: " << argv[0] << " <file_name> -fc <fault_coverage>\n";
        return 1;
    }
    CommonNs::TmUsage tmusg;
    CommonNs::TmStat stat;

    // Command-line arguments
    string file_name = argv[1];
    float FC = stof(argv[3]) / 100;

    // Map to store patterns and faults association
    unordered_map<int, vector<string>> faultPatternMap;
    unordered_map<string, vector<int>> patternFaultMap;
    // Save the final pruned patterns
    vector<int> patternPruned;

    // Read the fault information from the file
    ifstream file(file_name);
    if (!file)
    {
        cerr << "Error opening file.\n";
        return 1;
    }

```

```cpp
    int DS_Faults = 0;
    int totalFaults, DI_Faults, patternLength;
    // Read total faults and detected by implication faults
    file >> totalFaults >> DI_Faults >> patternLength;

    int currentPattern = -1;
    int patternDetectedNum[patternLength] = {};

    string line;
    while (getline(file, line))
    {
        // Check if it's a new pattern
        if (line[0] >= '0' && line[0] <= '9')
        {
            currentPattern++;
            continue;
        }

        istringstream iss(line);
        string ds, faultType, gateName;
        iss >> faultType >> ds >> gateName;

        patternDetectedNum[currentPattern]++;                              /
        string fault = faultType + " " + gateName;
        faultPatternMap[currentPattern].push_back(fault);                  /
        patternFaultMap[fault].push_back(currentPattern);   // Re
    }


    tmusg.periodStart();         // Record the function run time
```

Run Greedy Algorithm



```
tmusg.periodStart();        // Record the function run time

while(true){
    int maximum = 0;
    int maxPattern = -1;
    for (int i = 0; i < patternLength; i++){
        if(patternDetectedNum[i] > maximum){
            maximum = patternDetectedNum[i];
            maxPattern = i;
        }
    }
    if(maxPattern == -1){
        break;
    }
    for (const auto& fault : faultPatternMap[maxPattern]){
        for (const auto& patternToDelete : patternFaultMap[fault]) {
            if (patternToDelete == maxPattern) continue;
            faultPatternMap[patternToDelete].erase(remove(faultPatternMap[patternToDelete].begin(),
                                                        faultPatternMap[patternToDelete].end(), fault));
            patternDetectedNum[patternToDelete]--;
        }
        patternFaultMap[fault].clear();
        DS_Faults++;
    }
    patternPruned.push_back(maxPattern);
    patternDetectedNum[maxPattern] = 0; //don't need to check again
    if((float)(DS_Faults + DI_Faults)/totalFaults >= FC){
        break;
    }
}

tmusg.getPeriodUsage(stat);
cout <<"The total CPU time: " << (stat.uTime + stat.sTime) / 1000.0 << "ms" << endl;
cout <<"memory: " << stat.vmPeak << "KB" << endl; // print peak memory
```

Simple example (1)

pattern 0 detects

a sa1, b sa0, c sa0 faults

pattern 1 detects

c sa0, d sa0 faults

pattern 2 detects

b sa1, d sa0 faults

We can see that pattern 0 detects the most remaining faults

Pattern table	Detected faults
pattern 0	a sa1, b sa0, c sa0
pattern 1	c sa0, d sa0
pattern 2	b sa1, d sa0

Fault table	Patterns detect fault
a sa0	pattern 0
b sa0	pattern 0
b sa1	pattern 2
c sa0	pattern 0, pattern 1
d sa0	pattern 1, pattern 2

Simple example (2)

Pattern 0 is added to the pruned pattern list, and the faults detected by it (a sa1, b sa0, c sa0) would then be removed from the two tables.

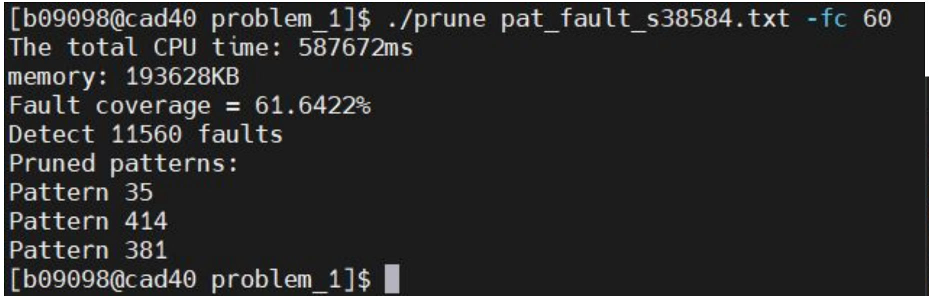
We can see that pattern 1 detects the most remaining faults now, thus would be our next target if needed.

Pattern table	Detected faults
pattern 1	d sa0
pattern 2	b sa1, d sa0

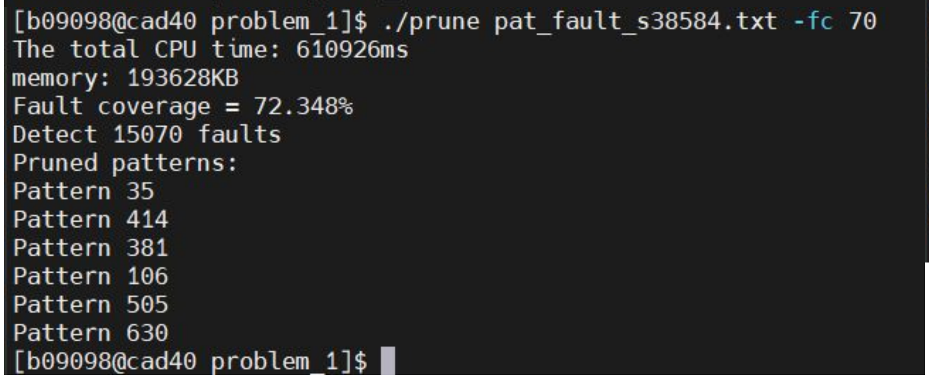
Fault table	Patterns detect fault
b sa1	pattern 2
d sa0	pattern 1, pattern 2

Results (all data acquired on 140.112.20.83)

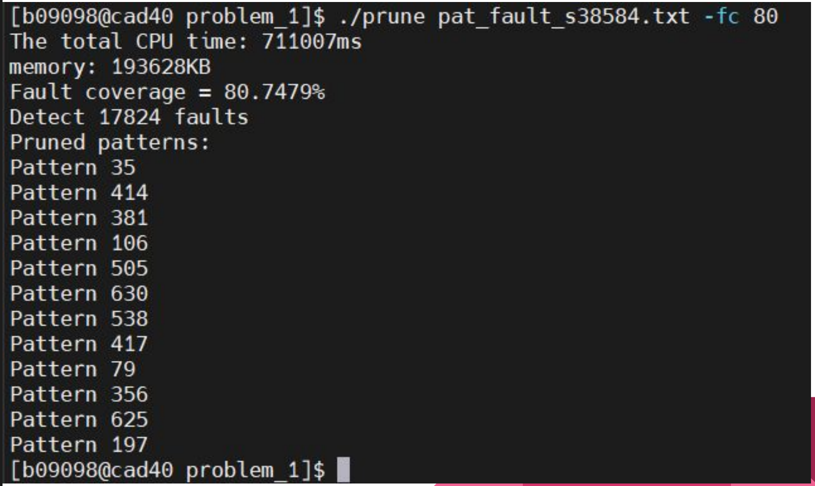
1. /Test_s38584/ (Origin patterns: 675)



```
[b09098@cad40 problem_1]$ ./prune pat_fault_s38584.txt -fc 60
The total CPU time: 587672ms
memory: 193628KB
Fault coverage = 61.6422%
Detect 11560 faults
Pruned patterns:
Pattern 35
Pattern 414
Pattern 381
[b09098@cad40 problem_1]$
```



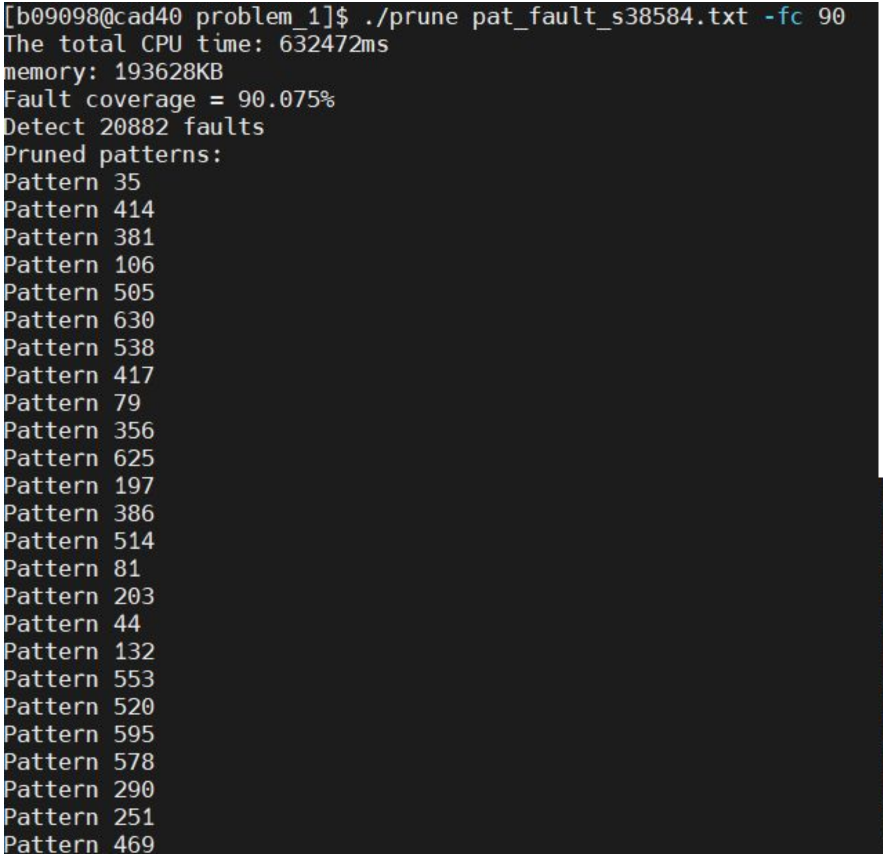
```
[b09098@cad40 problem_1]$ ./prune pat_fault_s38584.txt -fc 70
The total CPU time: 610926ms
memory: 193628KB
Fault coverage = 72.348%
Detect 15070 faults
Pruned patterns:
Pattern 35
Pattern 414
Pattern 381
Pattern 106
Pattern 505
Pattern 630
[b09098@cad40 problem_1]$
```



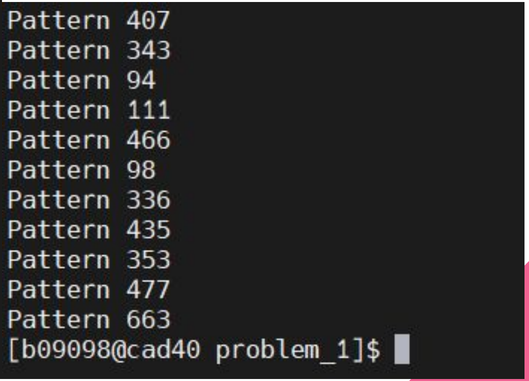
```
[b09098@cad40 problem_1]$ ./prune pat_fault_s38584.txt -fc 80
The total CPU time: 711007ms
memory: 193628KB
Fault coverage = 80.7479%
Detect 17824 faults
Pruned patterns:
Pattern 35
Pattern 414
Pattern 381
Pattern 106
Pattern 505
Pattern 630
Pattern 538
Pattern 417
Pattern 79
Pattern 356
Pattern 625
Pattern 197
[b09098@cad40 problem_1]$
```

s38584 cont.

90% FC : 36 patterns



```
[b09098@cad40 problem_1]$ ./prune pat_fault_s38584.txt -fc 90
The total CPU time: 632472ms
memory: 193628KB
Fault coverage = 90.075%
Detect 20882 faults
Pruned patterns:
Pattern 35
Pattern 414
Pattern 381
Pattern 106
Pattern 505
Pattern 630
Pattern 538
Pattern 417
Pattern 79
Pattern 356
Pattern 625
Pattern 197
Pattern 386
Pattern 514
Pattern 81
Pattern 203
Pattern 44
Pattern 132
Pattern 553
Pattern 520
Pattern 595
Pattern 578
Pattern 290
Pattern 251
Pattern 469
```

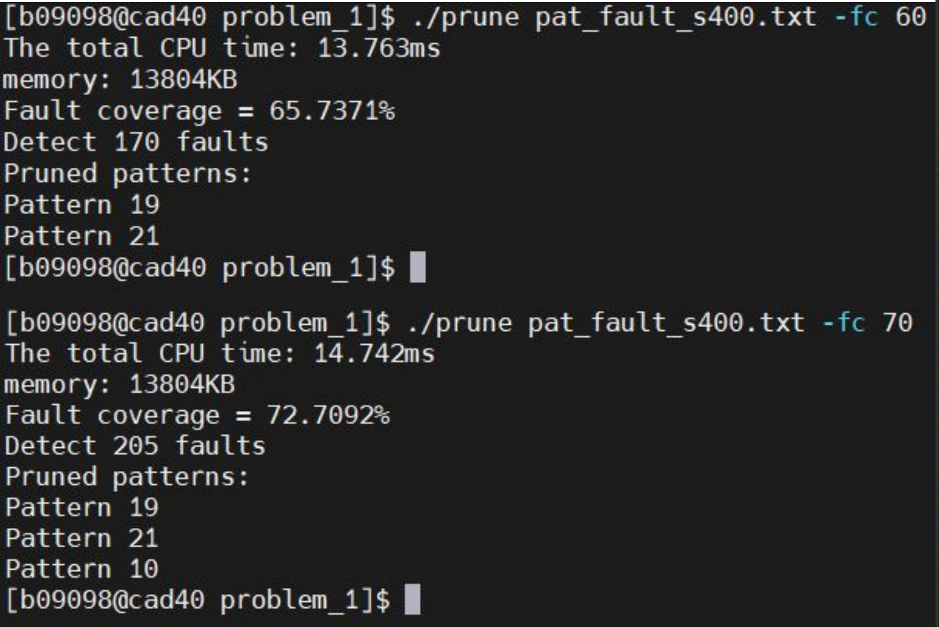


```
Pattern 407
Pattern 343
Pattern 94
Pattern 111
Pattern 466
Pattern 98
Pattern 336
Pattern 435
Pattern 353
Pattern 477
Pattern 663
[b09098@cad40 problem_1]$
```

Results

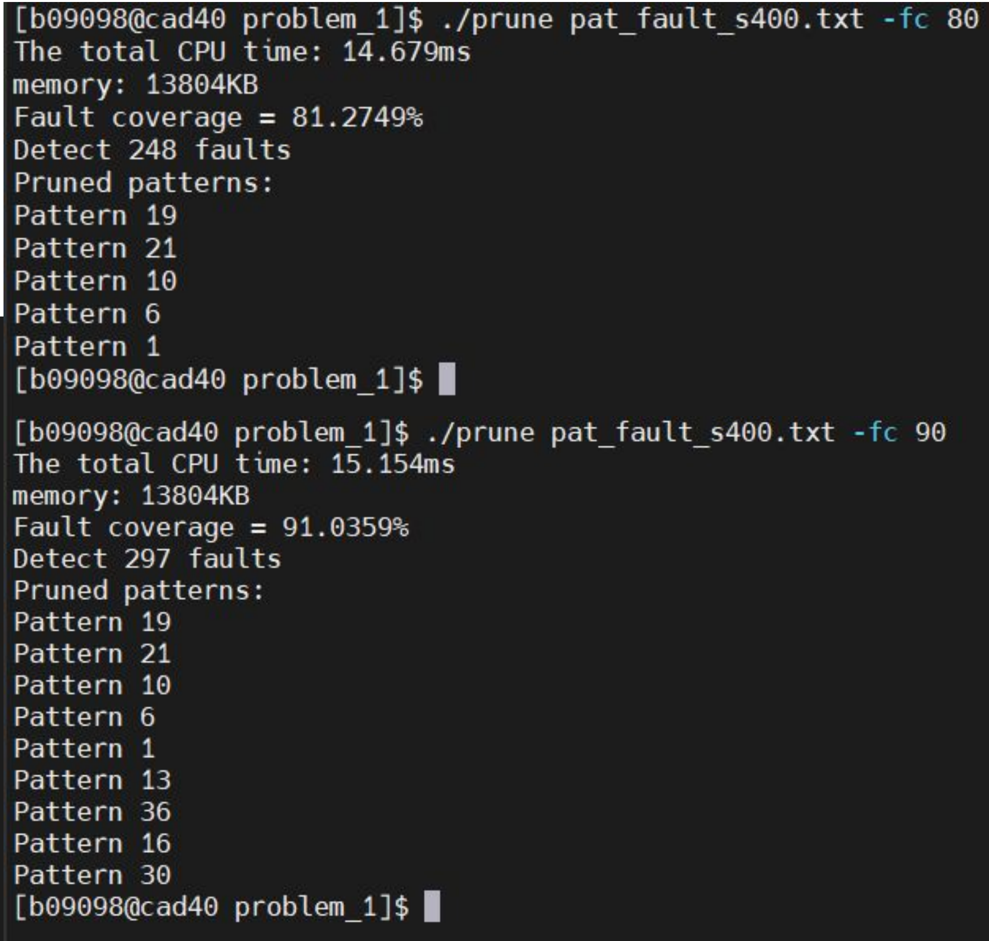
2. /Test_s400/ (Origin patterns: 39)

90% FC : 9 patterns



```
[b09098@cad40 problem_1]$ ./prune pat_fault_s400.txt -fc 60
The total CPU time: 13.763ms
memory: 13804KB
Fault coverage = 65.7371%
Detect 170 faults
Pruned patterns:
Pattern 19
Pattern 21
[b09098@cad40 problem_1]$
```

```
[b09098@cad40 problem_1]$ ./prune pat_fault_s400.txt -fc 70
The total CPU time: 14.742ms
memory: 13804KB
Fault coverage = 72.7092%
Detect 205 faults
Pruned patterns:
Pattern 19
Pattern 21
Pattern 10
[b09098@cad40 problem_1]$
```

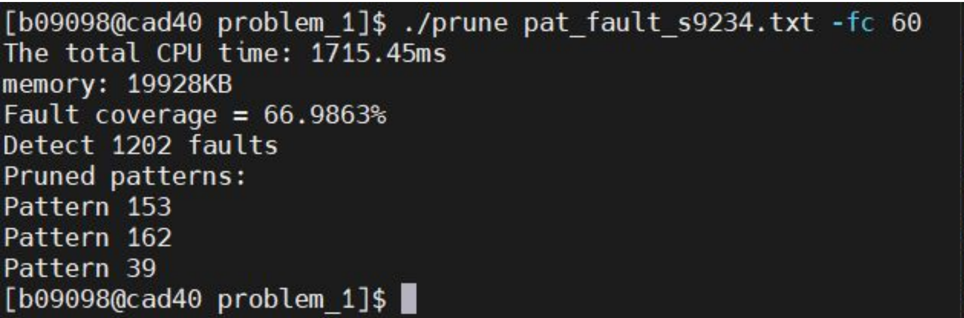


```
[b09098@cad40 problem_1]$ ./prune pat_fault_s400.txt -fc 80
The total CPU time: 14.679ms
memory: 13804KB
Fault coverage = 81.2749%
Detect 248 faults
Pruned patterns:
Pattern 19
Pattern 21
Pattern 10
Pattern 6
Pattern 1
[b09098@cad40 problem_1]$
```

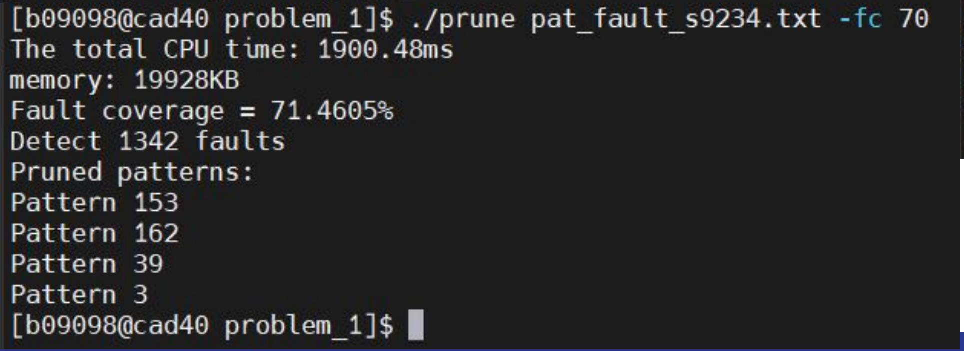
```
[b09098@cad40 problem_1]$ ./prune pat_fault_s400.txt -fc 90
The total CPU time: 15.154ms
memory: 13804KB
Fault coverage = 91.0359%
Detect 297 faults
Pruned patterns:
Pattern 19
Pattern 21
Pattern 10
Pattern 6
Pattern 1
Pattern 13
Pattern 36
Pattern 16
Pattern 30
[b09098@cad40 problem_1]$
```

Results

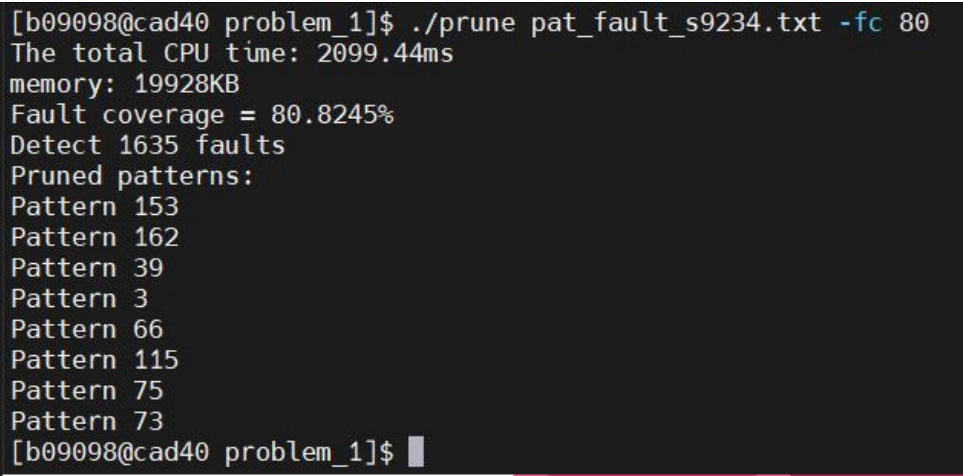
3. /Test_s9234/ (Origin patterns: 166)



```
[b09098@cad40 problem_1]$ ./prune pat_fault_s9234.txt -fc 60
The total CPU time: 1715.45ms
memory: 19928KB
Fault coverage = 66.9863%
Detect 1202 faults
Pruned patterns:
Pattern 153
Pattern 162
Pattern 39
[b09098@cad40 problem_1]$
```



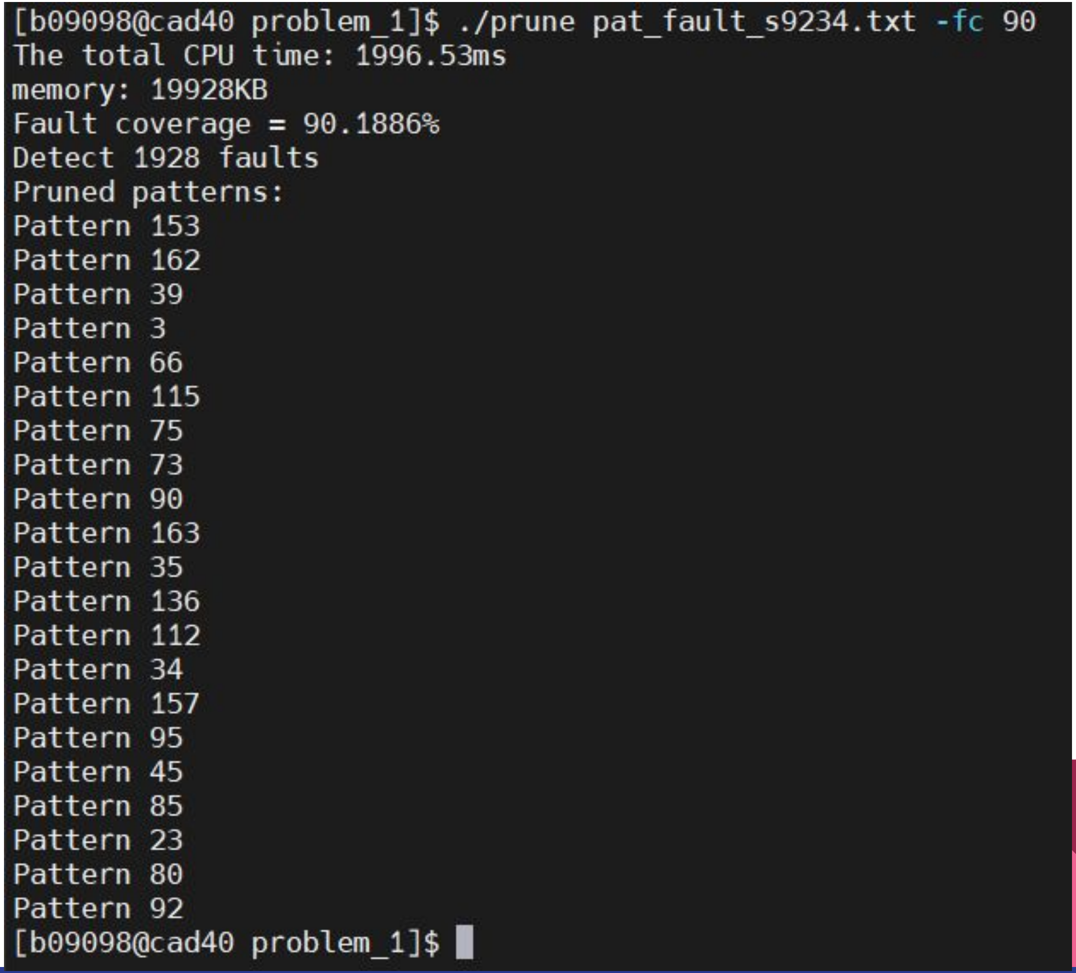
```
[b09098@cad40 problem_1]$ ./prune pat_fault_s9234.txt -fc 70
The total CPU time: 1900.48ms
memory: 19928KB
Fault coverage = 71.4605%
Detect 1342 faults
Pruned patterns:
Pattern 153
Pattern 162
Pattern 39
Pattern 3
[b09098@cad40 problem_1]$
```



```
[b09098@cad40 problem_1]$ ./prune pat_fault_s9234.txt -fc 80
The total CPU time: 2099.44ms
memory: 19928KB
Fault coverage = 80.8245%
Detect 1635 faults
Pruned patterns:
Pattern 153
Pattern 162
Pattern 39
Pattern 3
Pattern 66
Pattern 115
Pattern 75
Pattern 73
[b09098@cad40 problem_1]$
```

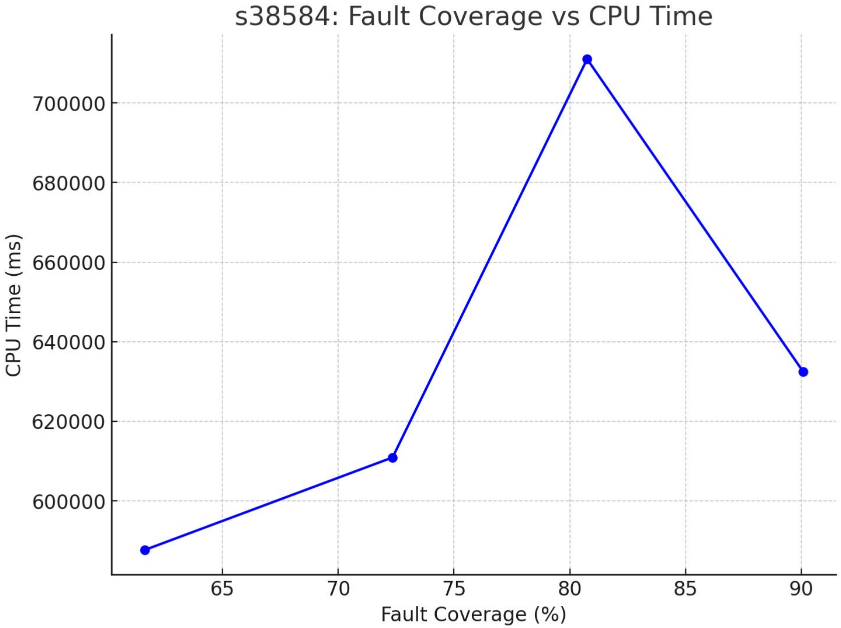
s9234 cont.

90% FC : 21 patterns

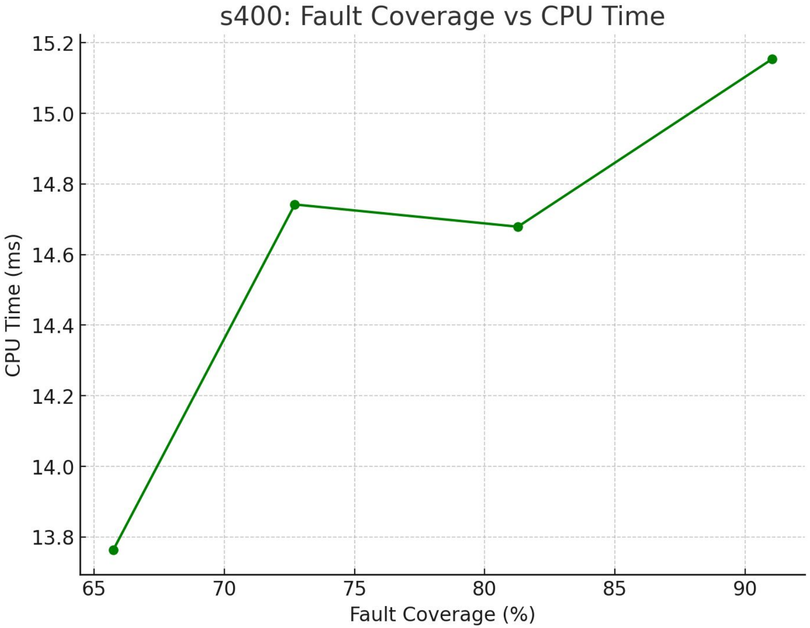


```
[b09098@cad40 problem_1]$ ./prune pat_fault_s9234.txt -fc 90
The total CPU time: 1996.53ms
memory: 19928KB
Fault coverage = 90.1886%
Detect 1928 faults
Pruned patterns:
Pattern 153
Pattern 162
Pattern 39
Pattern 3
Pattern 66
Pattern 115
Pattern 75
Pattern 73
Pattern 90
Pattern 163
Pattern 35
Pattern 136
Pattern 112
Pattern 34
Pattern 157
Pattern 95
Pattern 45
Pattern 85
Pattern 23
Pattern 80
Pattern 92
[b09098@cad40 problem_1]$
```

Conclusion

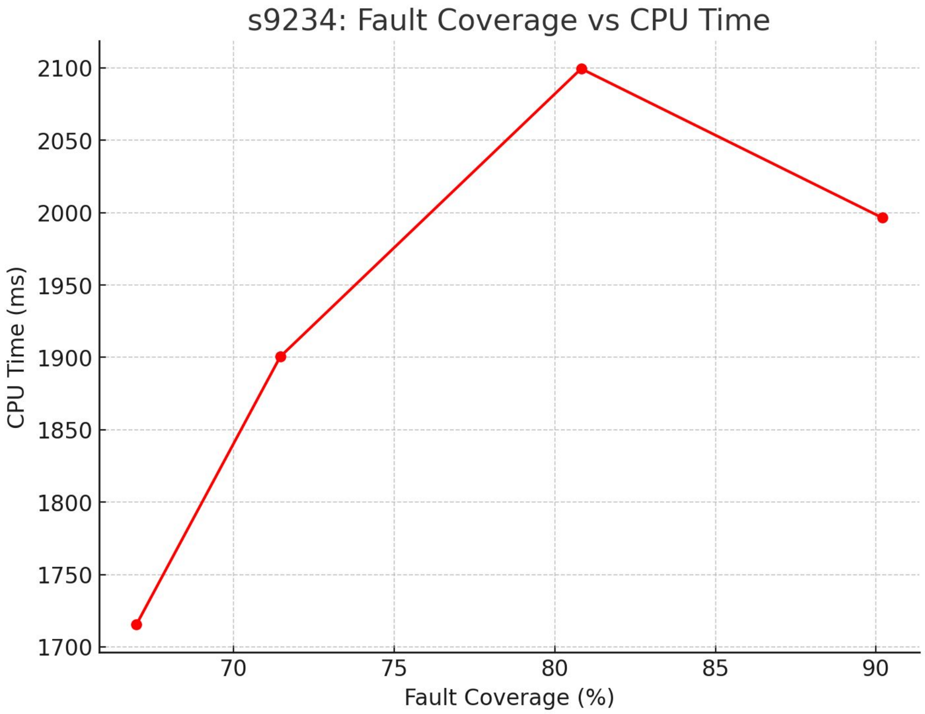


s38584: Fault Coverage vs CPU Time

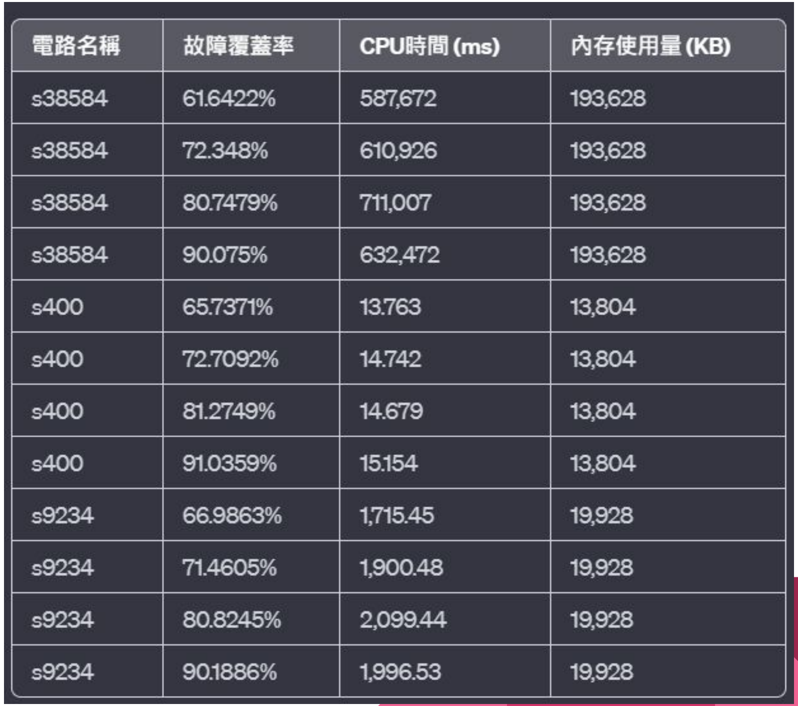


s400: Fault Coverage vs CPU Time

Conclusion



s9234: Fault Coverage vs CPU Time



電路名稱	故障覆蓋率	CPU時間 (ms)	內存使用量 (KB)
s38584	61.6422%	587,672	193,628
s38584	72.348%	610,926	193,628
s38584	80.7479%	711,007	193,628
s38584	90.075%	632,472	193,628
s400	65.7371%	13.763	13,804
s400	72.7092%	14.742	13,804
s400	81.2749%	14.679	13,804
s400	91.0359%	15.154	13,804
s9234	66.9863%	1,715.45	19,928
s9234	71.4605%	1,900.48	19,928
s9234	80.8245%	2,099.44	19,928
s9234	90.1886%	1,996.53	19,928

End of Report

Thanks for listening