FH-OÖ Hagenberg/ESD
Metrikorientierter Hardwareentwurf, WS 2015
Rainer Findenig, Markus Schutti © 2008 (R 2396)

4. Übung: PROL16: Automated Test Pattern Generation

Name(n): Punkte:

In der letzten Übung haben Sie mit Hilfe des Synopsys Design Compilers eine Scan-Chain in Ihren PROL16 eingebaut. Auf Basis dieser Scan-Chain sollen in dieser Übung mit Hilfe des ATPG-Werkzeugs Synopsys TetraMAX (`tmax`) Testvektoren für *Single-Stuck-At*-Fehler erstellt werden.

1 Theorie

Gute Einführungen in das Thema ATPG finden Sie beispielsweise in [Men08, Kapitel 2] und [Syn08b,
□ Kapitel 10]. Erklären Sie die folgenden Fehlerklassen:

- Detected
- Posdet (Mentor)/Possibly Detected (Synopsys)
- Untestable/Undetectable
- ATPG Untestable
- Undetected/Not Detected

2 ATPG in der Praxis

Machen Sie sich mit den Grundlagen des ATPG-Design-Flows vertraut [Syn08b, Kapitel 4]. Stellen Sie sicher, dass Sie alle benötigten Dateien zur Verfügung haben: die Netzliste und die STIL-Datei werden vom Design Compiler erzeugt, die Bibliotheken (in [Syn08b] als *Models* bezeichnet) stehen Ihnen im Verzeichnis `$AMS_DIR/verilog/c35b3/` zur Verfügung.

Erzeugen Sie nun die Testmuster für Ihren Entwurf. Das folgende Skript fasst die Schritte aus [Syn08b, Kapitel 4] zusammen:

```
read netlist netlist/prol16.v
read netlist $AMS_DIR/verilog/c35b3/*.v
report modules -summary
report modules -error
run build_model cpu
run drc prol16.spf
set faults -fault_coverage
set faults -summary verbose
```

```
add faults -all
set patterns internal
run atpg -random
report summaries > ../doc/atpg_results.rep
write pattern pfile.stil -format stil99 -replace
write faults ../doc/atpg_fault_list.rep -all -replace
```

Interpretieren Sie die Ergebnisse, die in die Datei `atpg_results.rep` ausgegeben werden! Beschreiben Sie in diesem Zusammenhang auch die beiden Coverage-Werte (siehe [Syn08b, Kapitel 10]). □

3 Simulation der Testmuster

TetraMAX erstellt in der Datei `pfile_stildpv.v` eine Verilog-DPV-Testbench [Syn08a] (DPV: *Direct Pattern Validation*), mit der Sie die erstellten Testvektoren simulieren können. Führen Sie eine Simulation mit dieser Testbench durch[1]. Beachten Sie, dass die erstellte Testbench eine PLI-Bibliothek verwendet und Sie die Simulation daher mit dem Parameter `-pli` starten müssen:

```
vlog ../syn/netlist/prol16.v
vlog ../syn/pfile_stildpv.v
vsim -pli /eda/etc/synopsys/libstildpv/libstildpv.so cpu_test
run -all
```

Genauere Informationen zum Erstellen dieser PLI-Bibliothek können Sie [Syn08a, Kapitel 2] entnehmen.

Erstellen und simulieren Sie nun zwei fehlerhafte Versionen der Netzliste: die erste Version soll einen Single-Stuck-At-Fehler enthalten, der in der Fehlerklasse *Detected by Simulation* liegt, die zweite einen nicht-detektierbaren Fehler. Dokumentieren Sie die Auswahl und die Vorgehensweise und interpretieren Sie die Ergebnisse. □

Literatur

[Men08] Mentor Graphics Corporation. *Scan and ATPG Process Guide*, 2008.

[Syn08a] Synopsys, Inc. *Test Pattern Validation User Guide*, September 2008.

[Syn08b] Synopsys, Inc. *TetraMAX ATPG User Guide*, September 2008.

[1] Wenn Sie die Signale Ihres DUTs im Wave-Fenster betrachten, können Sie anhand des Takts und des Eingangs `scan_enable_i` den Scan-Vorgang beobachten.

PROL16: Automated Test Pattern Generation
Übungsprotokoll zur Übung 4
Metrikorientierter Hardwareentwurf
Bernhard Selymes, Reinhard Penn, Robert Zeugswetter
07.12.2015

1 Theorie

Erklärung der Fehlerklassen:

- Detected
  Diese Klasse inkludiert alle Faults die vom ATPG erkannt werden. Eine Subkategorie ist Detected by Simulation, hier werden Testpatterns generiert und anschließend simuliert um sicherzustellen das mit diesem Testpattern Faults erkannt werden. Eine andere Subkategorie ist Detected by Implication. Faults dieser Art werden nicht von einem bestimmten Testpattern erkannt. Sie treten in der Scankette auf, zum Beispiel beim Scankettenteingang oder beim Takt der Scankette.

- Posdet
  (Mentor)/Possibly Detected (Synopsys) Diese Klasse ist in zwei Kategorien unterteilt. Die erste ist ATPG possibly detected. Hier sind alle Faults enthalten für die nicht bekannt ist ob sie am fehlerhaften Gerät den Wert 0 oder 1 annehmen. Mithilfe einer Analyse ist bewiesen, dass der Fault unter derzeitigen ATPG Bedingungen nicht zu 100% erkannt werden kann. Die Kategorie not analyzed-possibly detected besagt das selbe mit dem Unterschied das hier das Ergebniss der Analyse ob der Fault unter derzeitigen ATPG Bedingungen zu 100% erkannt werden kann inkonklusiv war.

- Untestable/Undetectable
  Faults in dieser Klasse können mit keinem Testpattern erkannt werden. Sie beinflussen allerdings auch das Verhalten des Systems nicht. Eine mögliche Art von Untestable Faults sind die Unused Faults, ein Fault dieser Art kann auftreten wenn der Fault an einem offenen Ausgang auftritt. Im Vergleich dazu gibt es auch noch die Klasse Tied, Faults in dieser Klasse treten auf Pins auf die auf einen bestimmten Wert gesetzt sind. Es kann zum Beispiel ein stuck-at-0 Fault auf einem Pin der auf 0 hängt nicht erkannt werden. Andere mögliche Arten sind Blocked und Redundant Faults.

- ATPG Untestable
  Faults in dieser Klasse können mithilfe von ATPG nicht erkannt werden, aber sie können durch andere Methoden erkannt werden, zum Beispiel funktionale Tests. Eine Art von Faults in dieser Klasse sind Faults von sequentiellen Elementen die nicht in der Scankette sind, zum Beispiel Latches.

- Undetected/Not Detected
  Faults kommen in diese Klasse wenn die Analyse nicht abgeschlossen oder abgebrochen wurde. Ein möglicher Grund dafür, ist das erreichen des ATPG Iterationslimit. Die zwei Subkategorien dieser Klasse sind not controlled und not observed. Not controlled Faults treten auf wenn der ATPG Algorithmus keinen 1 oder 0 erzeugen kann und der Status immer auf X bleibt. Not observed Faults treten auf wenn die Faults nicht in die Scankette oder an einen Ausgang weitergeleitet werden können.

2 ATPG in der Praxis

2.1 Ergebnisse

Die Ergebnisse der Simulation werden in der Datei in die unterschiedlichen Fehlerklassen unterteilt. Die test coverage ist naturgemäß größer als die fault coverage. Der Unterschied zwischen den beiden Werten kommt durch die undetectable faults zustande.

../atpg_results.rep

```
1      Uncollapsed Stuck Fault Summary Report
2   -----------------------------------------------
3   fault class                     code   #faults
4   ------------------------------  ----  ---------
5   Detected                         DT       7368
6     detected_by_simulation         DS      (5923)
7     detected_by_implication        DI      (1445)
8   Possibly detected                PT          0
9   Undetectable                     UD          6
10    undetectable-redundant         UR         (6)
11  ATPG untestable                  AU         19
12    atpg_untestable-not_detected   AN        (19)
13  Not detected                     ND          1
14    not-observed                   NO         (1)
15  -----------------------------------------------
16  total faults                                7394
17  test coverage                              99.73%
18  fault coverage                             99.65%
19  -----------------------------------------------
20             Pattern Summary Report
21  -----------------------------------------------
22  #internal patterns                           209
23      #basic_scan patterns                     209
24  -----------------------------------------------
```

2.2 Coverage

Beschreibung der Coverage-Werte laut TetraMAX® ATPG User Guide, Kapitel 10:

- Test Coverage
  Test coverage gives the most meaningful measure of test pattern quality and is the default coverage reported in the fault summary report. Test coverage is defined as the percentage of detected faults out of detectable faults.

- Fault Coverage
  Fault coverage is defined as the percentage of detected faults out of all faults. Fault coverage gives no credit for undetectable fault.

3 Simulation der Testmuster

3.1 Detected by Simulation

Der Ausgang eines Nands in der ALU wird auf 0 gesetzt. Dieser Fehler führt zu vielen Fehlern in der Simulation.

```
1   //detectable error
2   assign n148 = 0;
3   NAND30 U211 ( .A(n152), .B(n150), .C(alu_func_i[3]), .Q() );
```

Ausgabe Simulation:

```
# DPV: End of STIL data; validation of 209 patterns FAILED
with 343 output mismatches
```

3.2 Nicht-detektierbarer Fehler

In der ALU der Prol16 ist ein Ausgangsport nicht verbunden. An diesen wird ein Stuck-At-1 Fehler angehängt. Dieser Fehler wird nicht erkannt in der Simulation.

```
1   alu_16_DW01_add_0 add_110 ( .A({n2, side_a_i, n1}), .B({n2, add_b,
       add_cin}),
2          .CI(n2), .SUM({res_v_17, res_v_16, res_v_15, res_v_14, res_v_13,
3          res_v_12, res_v_11, res_v_10, res_v_9, res_v_8, res_v_7, res_v_6,
4          res_v_5, res_v_4, res_v_3, res_v_2, res_v_1,
              SYNOPSYS_UNCONNECTED__0})
5           );
6
7   // undetectable error
8   assign SYNOPSYS_UNCONNECTED__0 = 1;
```

Ausgabe Simulation:

```
# DPV: End of STIL data; validation of 209 patterns
completed successfully with no errors
```