FLOSS

User Guide

This document will describe how to use the public repo releasing the binary of the FLOSS (Functional Logic Obfuscation Security Assessment Suite) framework.

Requirements (Working Environment)

To run the jar files it is recommended that you have at least Java 1.8.

To use the bash script and run the m_refsm and redpen executables, it is recommended that you use a **linux operating system (ubuntu was used)**.

Running (Full Script)

There are 4 steps provided in this repo that enables extraction of the FSMs

1. Insert q pins into FFs (modify_netlist.jar)
2. Extract register interaction (redpen)
3. Extract all possible Flip Flop Groups (clique.jar)
4. Extract all FSMs (m_refsm)

Unless you have a good understanding of the jobs of each tool it is a good idea to use the provided script. It can be run using the following command,

```
$ bash scripts/recover_PIP2.sh <netlist_dir> <netlist_name>
<time_limit> <max_words> [library_file]
```

Below is an example run with the command used in bold,

```
$ bash scripts/recover_PIP2.sh netlist/des3_v3_8to6 obf_debug_final.v
10s 10
Working on obf_debug_final.v

Generating the new Netlist...
Name used is netlist/des3_v3_8to6/obf_debug_final
New netlist created!
```

```
Generating register dependency...
register dependency created at netlist/des3_v3_8to6/reg_dep.txt!

Generating cliques...
Cliques generated!

Generating fsms...
Note this is run with timeout so using Ctrl+c will not stop the
process
Error writen to log.err
scripts/recover_PIP2.sh: line 72: 31514 Killed                  (
timeout -s SIGKILL $TIME_LIMIT ./$BIN_DIR/m_refsm --lib
$FULL_LIB_PATH --net $NETLIST_DIR/$UPDATED_NETLIST_FILE --word
$NETLIST_DIR/fsms/all_ws.out --norst --tlimit 50000 $S_DOM >>
$LOG_FILE ) 2> $ERR_FILE_NAME
process crashed >_< (probably because of timeout)
Original FSMs generated.

Reading from netlist/des3_v3_8to6/fsms/ws_6.out
Writing to netlist/des3_v3_8to6/fsms/ws_6_log.out
Error written to netlist/des3_v3_8to6/fsms/ws_6_log.err
Note this is run with timeout so using Ctrl+c will not stop the
process
scripts/recover_PIP2.sh: line 82: 31519 Killed                  (
timeout -s SIGKILL $TIME_LIMIT ./$BIN_DIR/m_refsm --lib
$FULL_LIB_PATH --net $NETLIST_DIR/$UPDATED_NETLIST_FILE --word
$file --norst --tlimit 50000 $S_DOM > $NEW_FILE_NAME ) 2>
$ERR_FILE_NAME
process crashed (probably because of timeout)

Reading from netlist/des3_v3_8to6/fsms/ws_7.out
Writing to netlist/des3_v3_8to6/fsms/ws_7_log.out
Error written to netlist/des3_v3_8to6/fsms/ws_7_log.err
Note this is run with timeout so using Ctrl+c will not stop the
process
scripts/recover_PIP2.sh: line 82: 31525 Killed                  (
timeout -s SIGKILL $TIME_LIMIT ./$BIN_DIR/m_refsm --lib
$FULL_LIB_PATH --net $NETLIST_DIR/$UPDATED_NETLIST_FILE --word
$file --norst --tlimit 50000 $S_DOM > $NEW_FILE_NAME ) 2>
$ERR_FILE_NAME
process crashed (probably because of timeout)

Reading from netlist/des3_v3_8to6/fsms/ws_8.out
Writing to netlist/des3_v3_8to6/fsms/ws_8_log.out
Error written to netlist/des3_v3_8to6/fsms/ws_8_log.err
Note this is run with timeout so using Ctrl+c will not stop the
```

```
process
scripts/recover_PIP2.sh: line 82: 31529 Killed                  (
timeout -s SIGKILL $TIME_LIMIT ./$BIN_DIR/m_refsm --lib
$FULL_LIB_PATH --net $NETLIST_DIR/$UPDATED_NETLIST_FILE --word
$file --norst --tlimit 50000 $S_DOM > $NEW_FILE_NAME ) 2>
$ERR_FILE_NAME
process crashed (probably because of timeout)
```

Q Pin Insertion (Part 1)

Our tool uses the fact that an IC powers up into all logical 0s for the flip flops. The tool m_refsm (REFSM) does not have a distinction between Q and QN pins. REFSM has the ability to search for a reset state, but doing so is costly. It is ideal to give the tool Q pins and flagging REFSM with --norst when not looking for a reset state to ensure that the correct power on state is used.

The q pin insertion tool (modify_netlist.jar) reads in the netlist (given through the argument of the java command) and outputs a new version of the netlist as a file with a slightly modified name in the same location of the original netlist. Run using the following command,

```
java -jar bin/modify_netlsit.jar <netlist_file_name>
```

Note: *that the input netlist file needs to be verilog!*

An example run for the netlsit modification jar file can be seen below,

```
$ java -jar bin/modify_netlist.jar
netlist/des3_v3_8to6/obf_debug_final.v
Name used is netlist/des3_v3_8to6/obf_debug_final
```

The file written will be **`netlist/des3_v3_8to6/obf_debug_final_2.v`** where the file contains Q pins and removes netlist information not needed by the remaining tools in the workflow.

Register Interaction Extraction (Part 2)

The register interaction graph is used by the clique finding tool for more rapidly identifying cliques. It could have been included in the clique extraction tool, but the register interaction graph is needed for other workflows, and had already existed.

The register interaction graph extractor needs (for accuracy reasons) to know how the cell library works, which is why it needs an rlb file. We have been working on a verilog library file to

rlb file converter recently, but there is no expected release date. To run the register interaction graph use the following command,

```
./bin/redpen --lib <library_file_name> --net
<modified_netlist_file_name> --out <output_graph_name>
```

Below is an example of a redpen run,

```
$ ./bin/redpen --lib lib/harp.rlb --net
netlist/des3_v3_8to6/obf_debug_final_2.v --out
netlist/des3_v3_8to6/reg_dep.txt
Reading the library...
Done Reading the Library.
Number of Gates is 24452
Number of Regs is 470
Writing results to netlist/des3_v3_8to6/reg_dep.txt
Exiting...
```

The --lib flag says what library should be used. The library is an rlb library file type that describes the cell behavior. The --net flag says what netlist will be used. The --out flag says what file the output should be written to. The first 20 lines of the output file should look like the following,

```
n47791 --> n23912
n1691 --> n23912
n1688 --> n23912
n1693 --> n23912
n1692 --> n23912
n627 --> n23912
n47809 --> n23912
n23834 --> n23912
n47666 --> n23912
n23912 --> n23912
n24022 --> n23912
n47786 --> n23912
n47784 --> n23912
n47783 --> n23912
n23869 --> n23912
n24046 --> n23912
n47793 --> n23912
n23868 --> n23912
n620 --> n23912
n1155 --> n23912
```

Clique Extraction (Part 3)

The third step extracts all flip flop groups that form cliques. The current version will require a golden netlist as input that labels the OFSM/DFSM. The reason being that the maximum and minimum number of flip flops are not known by the tool. Future versions will allow the max and mins to be specified by options. To run the clique extractor use the following command,

```
java -jar clique <netlist> <register dependency> [Options]
```

The only option available is adjusting the maximum number of printed words in the files that are generated.

The program will create a folder in the directory of the netlist called fsms and add a file to it with the cliques found in the netlist. An example of running the program can be found below,

```
$ java -jar bin/clique.jar netlist/des3_v3_8to6/obf_debug_final_2.v
netlist/des3_v3_8to6/reg_dep.txt -max 100
"\DFSM0_CURR_STATE_reg"
"\DFSM6_CURR_STATE_reg"
"\DFSM7_CURR_STATE_reg"
"\OFSM3_CURR_STATE_reg"
"\DFSM8_CURR_STATE_reg"
"\OFSM5_CURR_STATE_reg"
"\OFSM1_CURR_STATE_reg"
"\DFSM11_CURR_STATE_reg"
"\DFSM13_CURR_STATE_reg"
"\DFSM14_CURR_STATE_reg"
"\DFSM18_CURR_STATE_reg"
"\OFSM0_CURR_STATE_reg"
"\DFSM1_CURR_STATE_reg"
"\DFSM2_CURR_STATE_reg"
"\DFSM3_CURR_STATE_reg"
"\DFSM4_CURR_STATE_reg"
"\DFSM5_CURR_STATE_reg"
"\DFSM16_CURR_STATE_reg"
"\DFSM9_CURR_STATE_reg"
"\DFSM19_CURR_STATE_reg"
"\OFSM2_CURR_STATE_reg"
"\OFSM4_CURR_STATE_reg"
"\DFSM10_CURR_STATE_reg"
"\DFSM12_CURR_STATE_reg"
"\DFSM15_CURR_STATE_reg"
"\DFSM17_CURR_STATE_reg"
Total REFSM state space is 4.101e+44
```

```
Total signals across all words is 175
Total number of words is 26
Max word size is 8
Min word size is 6
Writing to "netlist/des3_v3_8to6/fsms/all_ws.out"
Writing to "netlist/des3_v3_8to6/fsms/ws_6.out"
2353 words of size 6
Writing to "netlist/des3_v3_8to6/fsms/ws_7.out"
1943 words of size 7
Writing to "netlist/des3_v3_8to6/fsms/ws_8.out"
1344 words of size 8
The approximate state space for the clique method is 8.804e+5
Found 26 out of 26 words.
The number of cliques found is 5640.
```

Several files will be generated. The all_ws file will contain any possible fsm with equal probability up to the amount specified (or 100,000 if not specified number of FSMs are given). Other files will be in the fsms director and will have the name “ws_x.out”, where the x is an integer representing the number of flip flops in each FSM word.

FSM Extraction (Part 4)

The last part of the current workflow for extracting FSMs is the new multi-REFSM tool which extracts multiple FSMs from the same netlist in the order they are given. The command used for multi-REFSM is the following,

```
./bin/m_refsm --lib <library_file_name> --net
<modified_netlist_file_name> --word <word_file> [Options]
```

There are a lot of options for the m_refsm tool. Perhaps the best are the --norst and the --tlimit. The --norst flag will prevent multi-REFSM from looking for the reset signal. The --tlimit flag requires a number following it and will force multi-REFSM to kick out of state exploration for an individual FSM once a fixed number of transitions are found. If multi-REFSM kicks out using the --tlimit heuristics, a message will be printed to standard output (or the specified file if --out is used to redirect output to a file).

Below is an example of running the m_refsm tool,

```
$ ./bin/m_refsm --net netlist/des3_v3_8to6/obf_debug_final_2.v --lib
lib/harp.rlb --word netlist/des3_v3_8to6/fsms/all_ws.out --norst
--tlimit 50000
```

The output for an FSM looks like the following,

```
On state machine word_XXX

Looking for States...
Found States.
Completed YYY of ZZZ
Runtime estimation is TTT
```

word_XXX will be the name of the state machine. The YYY will be how many FSMs have been completed as of the time of the output. The value ZZZ will be the number of FSMs that were given to the m_refsm tool. The value TTT will be the expected runtime to complete the ***GIVEN*** FSMs.

Confidence Interval Generation (Part 5 Optional)

TODO