IWLS'93 Benchmark Set: Version 4.0

Ken McElvain

Mentor Graphics

15May93

1) Introduction

The purpose of IWLS benchmarks is to provide a way for researchers to evaluate their optimization techniques. To be useful the benchmarks need to have relevance to real designs. Since this is a moving target, the benchmarks need periodic updates. New benchmarks have been added to address new issues such as hierarchy, size, partitioning problems, level sensitive timing, gating of clocks and retiming.

The benchmarks are provided in a restricted EDIF format instead of blif or slif as in previous years. Finite state machines (FSMs) are additionally provided in kiss format, and two level and/or examples (PLAs) in espresso format. The benchmarks come with a C parsing library and three applications that use the library. The applications are an EDIF to BLIF or SLIF translator, a simulator for verifying optimization results, and a report generator that collects statistics about a netlist.

Each benchmark is accompanied by a test pattern file for verification which can be used with the included simulator. While this does not prove that the result of optimization satisfies the input netlist (which may contain don't cares), it does provide a reasonable level of checking. The test pattern files avoid the don't care conditions.

This release also includes an update of the target library. Sequential cells with asynchronous sets and resets were added to make it easier to include industrial sequential circuits. A few cells with multiple outputs were added to broaden the technology mapping opportunities.

2) Navigating About the Benchmark Release

```
iwls93/
        lib/                    - Target technology information.
                iwls93.e        - The EDIF description of the target library.
                iwls93.mis2lib- An incomplete target library description.
                iwls93.models - A complete target library description.
        testcases               - All test cases and test patterns.
                tv/             - One test pattern file for every EDIF file.
                edif/           - Every test case is available in EDIF.
                pla/            - Two level testcases are also available as espresso pla files.
                fsm/            - Finite state machines are also available as kiss files.
        src/                    - Source for the benchmark tools.
                Makefile        - Top level makefile for all tools.
                edifparse/      - Source for EDIF parser.
                netlist/        - Netlist data structures and action routines for parser.
                esim/           - EDIF netlist simulator.
                e2fmt/          - EDIF to {EDIF,SLIF,BLIF} format translator.
                erprt/          - Netlist report generator.
        bin/                    - Location where src Makefile installs tools.
        doc/                    - Documentation files.
                iwls93.ps       - Postscript version of this file.
```

3) Libraries

3.1) The Generic Library

The first library is a library of generic, technology independent gates. These generic gates are used in the representation of the benchmarks and in modeling of the target library cells. Gates included in the generic library are:

TRUE -	Outputs a logic 1
FALSE -	Outputs a logic 0
DC -	Outputs a don’t care
BUF -	Buffer
INV -	Inverter
TRI -	Tristate buffer
AND -	Arbitrary width AND gate
OR -	Arbitrary width OR gate
XOR -	Arbitrary width XOR gate
MUX2 -	2 way multiplexor
LATCH -	Transparent latch
LATCHS -	Transparent latch with asynchronous set

LATCHR -	Transparent latch with asynchronous reset
LATCHSR -	Transparent latch with asynchronous set and reset
DFF -	D flip-flop
DFFS -	D flip-flop with asynchronous set
DFFR -	D flip-flop with asynchronous reset
DFFSR -	D flip-flop with asynchronous set and reset

All the generic gates have active-high inputs. For the flip-flops and latches set and reset have priority over the clocked input. If both set and reset are on, then the output is a don't care.

3.2) The Target Library

The target library for this release of the benchmarks is based on the lib3 from IWLS'91. It has been extended with new cells implementing transparent latches and edge triggered flip-flops with asynchronous sets and resets. In accordance with what is usually found in ASIC libraries, these cells have complimentary outputs. A set of tristate drivers were added including inverting and performance versions. In addition, a few multi-output combinational cells were added, again along the lines of cells found in ASIC libraries. These cells include a half-adder, full-adder, and a 2-4 decoder. Lib1 and Lib2 from previous releases are not included in this release.

The target technology library is provided as an EDIF file called ".../iwls93/lib/iwls93.e" which contains a netlist for each technology gate. The netlist is built out of generic logic gates and is a valid logic model for the technology gate. Input pins of the logic model view have capacitance annotated as a property. Output pins have a maximum capacitance property, and an area property is annotated onto the logic model view. These values are used by a report generator described later.

In addition to the iwls93.e EDIF description a complete target library description is provided in the file iwls93.models. This file contains timing information for all the cells including multi-output and sequential cells. The format is defined in a comment at the top of the file.

The library is also included in the misII format. Cells not representable in this format were omitted from the misII format. This file is a corrected version of the lib3.mis2lib file release in '91. The corrections affect the timing data for four of the cells. QEO and QEN had incorrect input and output edge relationships. EO and EN had unreasonably large load coefficients.

4) The Benchmarks

All the benchmarks are available in EDIF format. Each EDIF file is accompanied by a test pattern file in the parallel "tv" directory that can be used to verify optimization results.

For those test cases where the original source was a pla, the equivalent pla may be found in the parallel directory "pla". All don't care information that was present in the pla file is also present in the EDIF netlist.

For those test cases that were originally an fsm, the equivalent kiss file may be found in the parallel directory “fsm”. Again, all don’t care information that was present in the fsm is also present in the EDIF netlist.

All benchmarks from the ‘91 benchmark set are included in this release. A few benchmarks have been moved to the directories “fsm.bad” and “edif.bad”. This was done when the benchmark as specified was provably uninitializable. When we say that a benchmark circuit is uninitializable, we mean that it lacks an initializing sequence. We do not mean that it won’t initialize in a simulator. Several of the good benchmarks will fail to initialize in a simulator. The simulator provided with this release has a pseudo-random initialization option to deal with such circuits.

5) Tools

The benchmarks come with a tool-set for reading and verifying the synthesis results. The tool-set includes a format converter, a simulator, a report generator, and a parsing library for linking in other tools. An important tool that is missing is a timing verifier.

If you invoke a tool without arguments, it will respond with a description of the available options and arguments.

All of the tools will directly read compressed input files.

5.1) E2fmt - An EDIF to other format converter.

E2fmt takes as input a set of EDIF netlist files and converts the view named user_lib.top.netlist to another format. Supported output formats are EDIF, SLIF, and BLIF. There are options for flattening the hierarchy to a single level netlist and for loading the file “.../iwls93/lib/iwls93.e”, which contains definitions of the target technology gates.

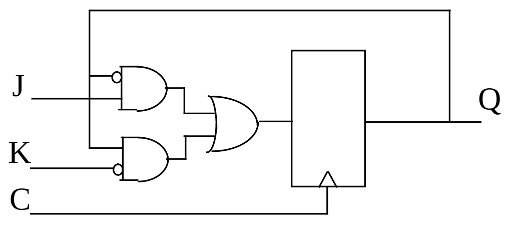
E2fmt is provided to maintain compatibility with users of previous releases of the benchmarks. One should note however that the BLIF and SLIF files produced will not be identical in detail to the originals and that this will likely lead to (hopefully small) differences in results for the older benchmarks that have been converted to EDIF and then back to BLIF.

5.2) Esim - An EDIF Unit Delay Simulator.

Esim is a simple event driven simulator that takes as input an EDIF netlist and a test pattern file. The input netlist is again expected to be named user_lib.top.netlist. This netlist is flattened down to the generic logic primitives, which esim has simulation models for, then the simulation is run, printing a trace of input and output values. The test pattern file contains both input stimulus and expected results. Any mismatches between the simulation and the expected results are flagged. The exit status indicates the presence of any mismatches. Esim has an option to suppress the trace for regression test purposes.

Esim is intended to give users of the benchmarks a relatively easy way to verify the results of their optimizations. The only requirement is that they be able to write a suitable EDIF netlist after optimization. Note that this verification is not a formal verification, but it will find most problems.

One issue that might be encountered is the initialization of state elements. There are a number of cases where one can prove that a given state element must initialize, but a logic simulator will conservatively assign an X. The figure below illustrates a circuit with this problem. If the DFF is currently in the X state and J=1, and K=0, then the JK circuit should set. Simulating a gate at a time, however, will leave the circuit stuck at X. To get around this problem, esim has an option to do random initialization of state elements. This option should be considered when esim reports a miscompare between a simulated X and an expected 1 or 0. It is recommended that esim be run several time with different random seeds if this option is used.



The test pattern file format used is very simple. It is free format with # to end of line being a comment. There are 4 commands, all terminated with a ';':

t <number>;	Advances time to the integer value <number>.
f <portname> [01ZX];	Force <portname> to have the given value.
q <clkname> [01ZX];	Force <clkname> to a value with priority over "f" commands at the same time.
c <portname> [01ZX];	Check that <portname> has the expected value.
e;	End the test pattern file.

The q command is used to avoid hold time problems between input assertions and clock changes.

5.3) Erprt - An EDIF based netlist report generator.

Erprt is a report generator that reads an EDIF netlist and reports statistics about that netlist. The report displays total cell area, a histogram of number of connections per net, a histogram of capacitance per net, and a list of capacitance limit violations.

6) Reporting

If your use of the benchmarks involves mapping into the provided technology, you should follow some measurement and reporting guidelines.

Primary input drive, and capacitance limits should be derived from the Q output of the FF primitive supplied in the target library. Primary output loading should have the same values as the D input load of the FF. A "legal" circuit should not violate the capacitance limits of the drivers of any nets in the netlist. Violations can be detected and reported by Erprt for your convenience.

Delay values reported should include the primary input drive values and the effects of the primary output loads. The circuit should be modeled as in a test jig made up of technology FF cells at the inputs and outputs.

Values of interest in a report are the instance count, number of nets, total gate area, average and peak number of connections per net, average and peak capacitance per net, and of course delay. Most of these values are available from Erprt.

7) Acknowledgments

Most of the benchmarks have been drawn from previous benchmark sets [yang91] and [lisanke89].

Conversion of benchmarks to the EDIF format and preparation of test pattern files for verification was done by David Rickel.

8) References

[yang91] Saeyang Yang, Logic Synthesis and Optimization Benchmarks User Guide: Version 3.0, Distributed as part of the IWLS91 benchmark distribution.

[lisanke89] Robert Lisanke, Logic Synthesis and Optimization Benchmarks User Guide: Version 2.0, Technical Report, Microelectronics Center of North Carolina, Dec. 1988.

[EDIF] EDIF Reference Manual Version 2.0.0 .