Journal of Electronic and Electrical Engineering
ISSN: 0976–8106 & E-ISSN: 0976–8114, Vol. 2, Issue 1, 2011, pp-24-29
Available online at http://www.bioinfo.in/contents.php?id=82

OPTIMIZATION OF BINARY AND MULTI-VALUED DIGITAL CIRCUITS USING MVSIS AND AIG REWRITING (ABC)

HEMANT DHABHAI[1], ABHISHEK KATARIYA[2], GEETAM TOMAR[3], YOGESH KRISHAN[4]
Department of Electronics & Comm., Sri Balaji College Of Engg. & Tech., Jaipur (Raj.), India, hemant1_anu@rediff.com
Department of Electronics & Comm., Sri Balaji College of Engg. & Tech., Jaipur (Raj.), India, abhishek_katariya@rediffmail.com
Director, MIR Lab, Gwalior (M.P.), India, gstomar@ieee.org
Department of Electronics & Comm., Sri Balaji College of Engg. & Tech., Jaipur (Raj.), India, yogi_sikar@yahoo.com
*Corresponding Author: Email- abhishek_katariya@rediffmail.com

Received: August 24, 2011; Accepted: September 08, 2011

Abstract– Benchmark designs are the basis for the performance evaluation of today's EDA tools for assisting the synthesis of an integrated circuit. The most commonly employed method to verify the competitiveness of a new tool consists of applying this tool to a set of benchmark designs in a given experimental setting. The experimental results are then compared to those obtained by applying a comparable state-of-the-art tool to the same set of designs. Berkeley Logic Interchange Format (BLIF) is used to represent combinational and sequential logic networks used in academic logic synthesis and verification tools.
This paper is concerned with focusing the comparison of AIG rewriting in ABC with logic synthesis in MVSIS on MCNC benchmark. We have also performed Node reduction with MVSIS (script.rugged) and ABC (resyn2), and also shows area and delay reduction by mapping of optimize benchmark. We have found out that AIG rewriting is an innovative technique for combinational logic synthesis.
Key words – MVSIS, MCNC benchmark, AIG, ASIC, BLIF, EDA tools

INTRODUCTION

Logic synthesis addresses the problem of translating a register-transfer level description of a design into an optimal logic–level representation. Very large scale Integration technology is in wide use in modern digital systems. VLSI technology currently allows hundreds of thousands of transistors in a single application-specific integrated circuit (ASIC), and the level of integration is increasing at a rapid rate [1].
Many tools are used to measure the quality and correctness of these circuits before fabrication for assisting the synthesis of an integrated circuit. They include reduced design time, reduced probability of design error and also reduce the area, delay and power of a particular circuit. MVSIS is a logic synthesis system, which supports the data structures and procedures needed for technology
independent binary and multi-valued (MV) logic synthesis [2, 3]. Similarly, AIGs can be used to represent arbitrary Boolean formulas and circuits, and are implemented in several logic synthesis and verification systems. It is possible to represent every basic logic operation by AND gates and inverters and therefore by AIGs.
Improving verification productivity and avoiding respins have lead to a structured, design-for-verification methodology. In the past decades, many functional verification tools and methodologies have been developed, including simulators, formal verifiers, and debugging tools. Among these verification methods, SIS, MVSIS and ABC has become the mainstream methodology for functional verification to generate as large a representative set of scenarios for a given digital circuits as possible under project constraints.

MVSIS

Logic synthesis exemplified SIS and MVSIS by applying a sequence of optimization steps i.e. SWEEP (For removing nodes), ELIMINATE and RESUBSTITUTE (For finding better boundary), FAST_EXTRACT (For discovering better logic boundary) and SIMPLIFY and FULL_SIMPLIFY (For simplifying the node representation)[12]. MVSIS is a program modeled after SIS, but the logic network it works on is such that all variables can be multi-valued, each with its own range. MVSIS can read and write BLIF.

MVSIS input formats can be-

1. PLA or BLIF for Binary functions and networks
2. BLIF-MV for Multi-valued functions and networks
3. And for, FSMs and finite automata three options are available
   - Using BLIF/BLIF-MV followed by "*stg_extract*"
   - Using modified KISS2 format
   - Using modified BLIF-MV format

To optimize any type of digital circuit, we can perform following steps:-

1. Node simplification

2. Algebraic Decomposition
3. Pairing and encoding
4. Network manipulation

BERKELEY LOGIC INTERCHANGE FORMAT i.e. BLIF

A Binary Valued/ Multi Valued (BV/MV) circuit can be input to MVSIS as a net list of BV-node (command: read_blif or read_blif_mv) MV-nodes (command: read_blif_mv). This BLIF format can be generating by using VIS/VL2MV, by the Verilog front-end to VIS (vl2mv) or can be written out by VIS.

The Berkeley Logic Interchange Format (BLIF) is used to represent combinational and sequential logic networks used in academic logic synthesis and verification tools. BLIF became popular with SIS and is currently supported in VIS, MVSIS, and most recently in ABC . BLIF is also supported by the academic physical design tool VPR [9].

BLIF-Format

```
.model <you name it>
.inputs <list of input names>
.outputs <list of output names>
.names <list of fan-in names> <node_name>
<PLA representation of the node's function>
.names <list of fanin names> <node_name>
<PLA representation of the node's function>
                                    …
.latch <latch_input> <latch_output> <reset_value>
.latch <latch_input> <latch_output> <reset_value>
…
.end
```

Fig. 1-BLIF Format

ABC (AIG REWRITING)

An And-Inverter Graph (AIG) is, a directed acyclic graph (DAG), or a network which have two type of node i.e. AND and INVERTER. AIG is not canonical. A node with no incoming edges is a primary input (PI). A node with two incoming edges is a two-input AND gate. An edge is either complement or not. A complement edge indicates the inversion of the signal[6,9]. Certain nodes are marked as primary outputs (POs). Register if present are considered as PI/PO pairs. Two Boolean functions, F and G, belong to the same NPN-class (Or equivalent NPN) if F can be derived from G by negating (N) and permuting (P) inputs and negating (N) the output.

Example: Functions F = ab + c and G = ac + b are NPN-equivalent because swapping b and c make them identical. Function F = ab + c and G = ab are not NPN-equivalent because no amount of permuting and complementing variables can make a 3-variable function equivalent to a 2-variable function.

TYPES OF AIGS

COMBINATIONAL AIGS

A combinational And-Invertor Graph (AIG) is a Boolean network composed of two-input ANDs and inverters. To derive an AIG, the SOPs of the nodes in a logic network are factored, the AND and OR gates of the factored forms are converted into two-input ANDs and inverters using DeMorgan's rule, and these nodes are added to the AIG manager in a topological order.

The size (area) of an AIG is the number of its nodes; the depth (delay) is the number of nodes on the longest path from the PIs to the POs. The goal of optimization by local transformations of an AIG is to reduce both area and delay.

SEQUENTIAL AIGS

Sequential AIGs extend combinational AIGs with technology-independent D-flip-flops with one input and one output, controlled by the same clock, omitted in the AIG representations [6].

We represent flip-flops in the AIG explicitly as additional PI/POs pairs. The PIs and register outputs are called *combinational inputs* (CIs) and the POs and register inputs are called *combinational outputs* (COs). The additional pairs of CI/CO nodes follow the regular PIs/POs, and are in one to one correspondence with each other [10].

The AIG is a Boolean network composed of two types of nodes: two-input AND-gates and inverters.

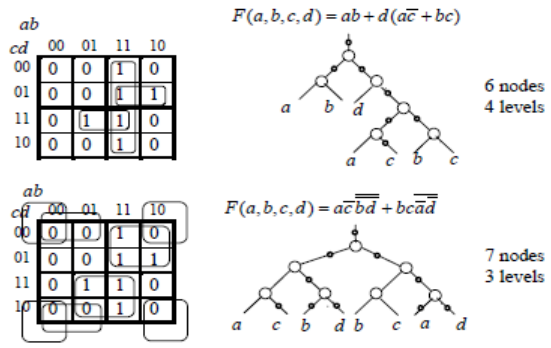

Fig. 2-Two different AIGs for a Boolean function [14]

A representation of a Boolean function is canonical if, for any function, there exists only one representation of this type. AIGs are not canonical, that is, the same function can be represented by two functionally equivalent AIGs, which have different structure. An example of such function is shown in Fig. (2) [14]

OUR APPROACH

To address the problems mentioned above, we have tested several benchmark circuits like Duke2, Rd84, Misex2, B12, Cordic, Cps, Pdc, Spla, C432, B9, Dalu, K2 etc. on MVSIS and ABC rewriting for optimization of MV networks.

OPTIMIZATION OF BENCHMARK BY MVSIS

To analyze the performance of this tool, script.rugged is applied over 19 combinational MCNC benchmark circuit.structure of
script.rugged is shown in figure 1.

```
fullsimp -m nocomp
sweep;
eliminate  -1
simplify -m nocomp
eliminate  -1
sweep;
eliminate  5
```

```
simplify -m nocomp
resub
resub ;
sweep
eliminate ;
sweep
fullsimp -m nocomp
```

Fig. 3- script.rugged

In the Table 1, the first column shows the standard MCNC benchmark circuit. Next seven columns show the statics of the benchmark circuit before applying the standard script.rugged. Here, PI is the number of primary inputs; PO is the number of primary output, while Lits indicate the literals. The next section of the table shows, the reduced number of node, level, cubes, literals and literal(ff) after run MVSIS (script.rugged) over the benchmark.

Example: Script.rugged for Duke2 benchmark circuit

```
read_blif duke2.blif
print_stats -s
read_library mcnc.genlib
map
print_map_stats
fullsimp -m nocomp
map
sweep;
eliminate
simplify -m nocomp
eliminate
sweep;
eliminate
simplify -m nocomp
resub
resub ;
sweep
eliminate ;
sweep
fullsimp -m nocomp
map
print_map_stats
print_stats –s
```

DEGREE OF REDUCTION IN NODE AND LITERAL

Table-2 shows the degree of reduction in benchmark circuit. Here B12 circuit have maximum degree of reduction in node as well as literal is 0.857 and 0.850 respectively. In some of the benchmarks, i.e. PDC, SPLA, DES and K2, number of node and literals are increased, but area is reduce as shown in the Table-2.

OPTIMIZATION BY ABC

AIG rewriting is implemented in the sequential logic synthesis and verification system, ABC, as commands rewrite, refactor and balance. A rewriting script, resyn2, has 10 passes over the network as follows : b; rw; rf; b; rw; rwz; b; rfz; rwz; b. In the abbreviation notation, b (balance) stands for AIG balancing, rw/rf (rewrite/refactor) stands for AIG rewriting/refactoring, and rwz/rfz is the same but with zero-cost replacement allowed.

The resyn2 script optimizes area under delay constraints. It starts by balancing to reduce delay upfront as much as possible. Next, rewriting/refactoring and balancing are interleaved. During this, rewriting/refactoring tries to reduce area while not increasing area. Zero-cost replacements are enabling later in the script to facilitate creating new rewriting opportunities. This resyn2 is stopped after three iterations.

DEGREE OF REDUCTION IN MVSIS AND ABC

This section compares AIG rewriting in ABC with logic synthesis in MVSIS on MCNC benchmark.

Table 4 shows that Node reduction with MVSIS (script.rugged) and ABC (resyn2), and also shows that area and delay reduction of mapping of optimize benchmark. ABC (resyn2) based on area optimization under delay constraints. Therefore in some benchmarks delay is increased as compare to MVSIS (script.rugged).

CONCLUSION

After comparing AIG rewriting in ABC with logic synthesis in MVSIS on MCNC benchmark. We conclude that AIG rewriting is an innovative technique for combinational logic synthesis. This experiment shows that AIG rewriting often leads to quality comparable or better than those afforded by the logic synthesis script in MVSIS. Table 4 shows that Node reduction with MVSIS (script.rugged) and ABC (resyn2), and also shows that area and delay reduction of mapping of optimize benchmark.

References

[1] "Innovative Verification and Synthesis Techniques for Achieving High-Quality SoC Designs" Hong-Zu Chou, Department of Electrical Engineering ,College of Electrical Engineering and Computer Science, National Taiwan University, June 2010

[2] "Logic Synthesis for VLSI Design" Richard L Rudell, University of California, Berkley, California

[3] "Logic Synthesis and Optimization Benchmarks User Guide- Version 3.0" - Saeyang Yang

[4] "Optimizing behavioral transformations using taylor expansion diagrams' A Dissertation Presented by QIAN REN, University of Massachusetts Amherst

[5] "Optimization of Multi-Valued Multi-Level Networks"- M.Gao, J-H. Jiang, Y. Jiang, Y. Li, A. Mishchenko, S. Sinha, T. Villa and R. Brayton; 32nd IEE International Symposium on Multiple-Valued Logic (ISMVL'02)

[6] "DAG-Aware AIG Rewriting – A Fresh Look at combinational Logic Synthesis"– Alan Mishchenko, Satrajit Chatterjee, Robert Brayton; DAC 2006

[7] 'Minimization of Multiple Valued Functions in Post Algebra' – Elena Dubrova, Yunjian Jiang, Robert Brayton

[8] "Multi-Valued Logic Synthesis" Robert K Brayton, Sunil P Khatri

[9] "Quick Look under the Hood of ABC – A Programmer's Manual", December 25, 2006

[10] Berkeley Logic Synthesis and Verification Group, University of California, Berkeley, "ABC: A System for Sequential Synthesis and Verification",

[11] Berkeley Logic Synthesis and Verification Group, University of California, Berkeley, “Berkeley Logic Interchange Format (BLIF)”
[12] Donald Chai, Jie-Hong Jiang, Yunjian Jiang, Yinghua Li, Alan Mishchenko, Robert Brayton “MVSIS 2.0 User’s Manual” , University of California, Berkeley CA 94720
[13] Ramakanth Kondagunturi, Eugene Bradley, Kristi Maggard, and Charles Stroud, “Benchmark circuits for analog and mixed-signal string”, IEEE paper, V. Betz, et al., University of Kentucky
[14] "ABC: An Academic Industrial-Strength Verification Tool", Robert Brayton, Alan Mishchenko, EECS Department, University of California, Berkeley, CA 94720, USA

Table 1- *Statics of benchmark circuit before and after applying the mvsis (script.rugged)*

Bench-Mark Circuit	Circuit Statics							After Run MVSIS (script.rugged)				
	PI	PO	Node	Level	Cubes	Lits	Lits(ff)	Node	Level	Cubes	Lits	Lits(ff)
Duke2	22	29	385	8	590	992	992	362	8	772	898	890
Rd84	8	4	495	9	894	1442	1442	495	10	1115	1278	1249
Misex2	25	18	123	6	134	246	246	83	5	156	189	186
B12	15	9	769	9	1214	1860	1860	110	7	229	279	269
Cordic	23	2	2079	12	3283	4969	4969	1500	13	3414	4163	3909
Cps	24	109	1219	9	1756	3022	3022	1117	9	2277	2808	2703
Pdc	16	40	2326	17	4203	8259	7364	3528	14	7821	10046	9140
Spla	16	46	2237	18	4128	7961	7034	3424	13	7393	9173	8578
C432	36	7	355	29	373	567	567	173	19	341	442	390
C1355	41	32	949	44	949	1467	1467	220	10	428	684	678
C1908	33	25	798	55	798	1245	1245	274	18	566	795	771
C2670	233	64	1467	46	1467	2195	2195	477	13	873	1137	1088
C6288	32	32	4800	246	4800	7184	7184	*	*	*	*	*
T481	16	1	1075	11	1555	2673	2673	783	11	1945	2314	2142
Rot	135	107	731	20	1024	1998	1998	602	13	1162	1427	1347
B9	41	21	275	20	351	408	408	98	6	175	220	204
Dalu	75	16	2609	59	3595	4541	4541	743	15	1507	1962	1805
Des	256	245	2263	10	3957	8991	8991	3823	12	7758	9864	9453
K2	45	45	399	4	1521	3176	3176	1368	10	2850	3409	3366

Table 2- *Degree of reduction in node and literal*

BenchMark Circuit	Circuit Statics		After MVSIS (Script.rugged)		Degree of reduction	
	Node	Literal	Node	Literal	Node	Literal
Duke2	385	992	362	898	0.060	0.095
Rd84	495	1442	495	1278	0	0.114
Misex2	123	246	83	189	0.325	0.231
B12	769	1860	110	279	0.857	0.850
Cordic	2079	4969	1500	4163	0.278	0.162
Cps	1219	3022	1117	2808	0.084	0.071
Pdc	2326	8259	3528	10046	(-)0.517	(-)0.216
Spla	2237	7961	3424	9173	(-)0.531	(-)0.152
C432	355	567	173	442	0.513	0.220
C1355	949	1467	220	684	0.768	0.534
C1908	798	1245	274	795	0.657	0.386
C2670	1467	2195	477	1137	0.675	0.482
C6288	4800	7184	*	*	*	*
T481	1075	2673	783	2314	0.272	0.134
Rot	731	1998	602	1427	0.176	0.286
B9	275	408	98	220	0.644	0.461
Dalu	2609	4541	743	1962	0.715	0.562
Des	2263	8991	3823	9864	(-)0.689	(-)0.097
K2	399	3176	1368	3409	(-)2.429	(-)0.073

Table 3- *Optimization of Benchmark by ABC on Rewriting Performance*

Bench Mark Circuit	First Iteration (rwz)			Second Iteration (rwz)			Third Iteration (rwz)		
	Node Rewritten	Gain	% Gain	Node Rewritten	Gain	% Gain	Node Rewritten	Gain	% Gain
Duke2	274	57	10.59	221	30	6.24	203	19	4.21
Rd84	224	71	9.97	213	30	4.68	201	22	3.60
Misex2	45	12	10.17	36	4	3.77	36	1	0.98
B12	632	246	25.13	401	114	15.55	342	67	10.82
Cordic	1164	1785	64.37	310	125	12.65	242	69	8.00
Cps	948	251	15.54	625	73	5.35	578	40	3.10
Pdc	2397	208	4.16	2082	208	4.34	1930	116	2.53
Spla	2195	165	3.46	1904	219	4.76	1767	109	2.49
C432	99	31	14.62	65	14	7.73	48	9	5.39
C1355	220	100	19.53	59	18	4.37	54	0	0.0
C1908	125	31	7.54	74	8	2.11	70	2	0.54
C2670	169	78	12.07	102	8	1.41	95	0	0.0
C6288	*	*	*	*	*	*	*	*	*
T481	521	514	37.11	262	121	13.89	193	55	7.33
Rot	559	299	27.06	272	67	8.31	214	36	4.87
B9	36	28	23.14	25	7	7.53	19	2	2.33
Dalu	714	571	32.91	311	53	4.55	249	14	1.26
Des	2003	993	20.99	1128	245	6.55	842	47	1.35
K2	1470	458	20.01	1055	104	5.68	921	38	2.20

Table 4- *Comparison between MVSIS and ABC*

Bench Mark Circuit	(Degree of) Reduction in MVSIS			(Degree of) Reduction in ABC		
	Node	Area	Delay	Node	Area	Delay
Duke2	0.060	0.084	0.0	0.099	0.136	(-)0.1
Rd84	0	0.131	(-)0.5	0.027	0.147	(-)1.6
Misex2	0.325	0.093	(-)0.4	0.036	0.077	0.4
B12	0.857	0.820	2.0	0.408	0.434	0.2
Cordic	0.278	0.386	(-)1.2	0.422	0.537	(-)0.4
Cps	0.084	0.104	(-)0.8	*	*	*
Pdc	(-)0.517	0.100	0.3	0.114	0.094	0.8
Spla	(-)0.531	0.098	0.1	0.119	0.113	0.3
C432	0.513	(-)0.037	(-)1.9	0.178	0.141	1.4
C1355	0.768	0.034	(-)0.3	0.083	0.058	1.5
C1908	0.657	0.009	1.1	0.027	0.043	0.2
C2670	0.675	0.0805	(-)0.8	*	*	*
C6288	*	*	*	*	*	*
T481	0.272	0.149	0.0	0.182	0.221	(-)0.3
Rot	0.176	0.282	2.3	*	*	*
B9	0.644	0.105	0.5	0.121	0.161	0.8
Dalu	0.715	0.347	6.6	0.221	0.259	3.6
Des	(-)0.689	0.158	2.1	0.120	0.123	0.5
K2	(-)2.429	0.065	(-)0.3	0.124	0.147	0.3