SEQUENTIAL AND COMBINATIONAL EQUIVALENCE CHECKING

Nikolay Pavlovich Laptev
Department of Computer Science – UCLA
Kunal Bindal
Department of Computer Science – UCLA

ABSTRACT
Sequential and combinational equivalence checking both are quite important in circuit design synthesis. **[give example of sequential synthesis].** In combinational synthesis we are given two circuits and are asked weather or not they are equivalent. In this paper we show how to improve the speed of combinational equivalence checker (CEC) by using more intelligent simulation as well as some heuristics. Also CEC work much worse when the two circuits are not equivalent, we show how to counter this problem. **[Add results summary]**

1 INTRODUCTION+BACKGROUND

Combinational equivalence checking works by taking two circuits and checking if they produce the same output. In this paper, we will show how a modern CEC equivalence engine can be modified to output an equivalence formula.

Generally when invoking a modern CEC equivalence checker it generates a *miter*. A miter is a circuit which is derived from the two other circuits which need to be checked for equivalence. This is done by adding 2-input XOR gates on top of corresponding outputs and connecting the outputs of these XOR gates to a large OR gate. The miter is a single output of this OR gate. If the output can be shown to be 0, then the two circuits are equivalent).

Clearly this can be posed as a SAT problem, and if the SAT problem can be proved to be unsatisfiable, the resolution proof generated by the SAT solver will be enough to prove that the two circuits are equivalent.

Consider the following four clauses:

$$1.(a \vee \bar{b})$$

$$2.(\bar{a} \vee c \vee d)$$

$$3.(b \vee \bar{c})$$

$$4.(\bar{d})$$

Then the resolution proof which shows unsatisfiability would be:

$$1p.(\bar{b} \vee c \vee d)[from-1\ \&\ 2]$$

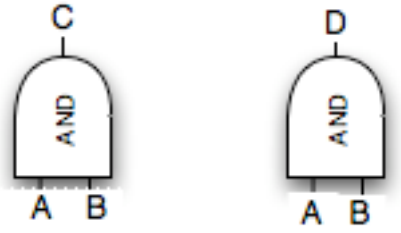
$$2p.(d)[from-1p\ \&\ 3]$$

$$3p.()[from-2p\ \&\ 4]$$

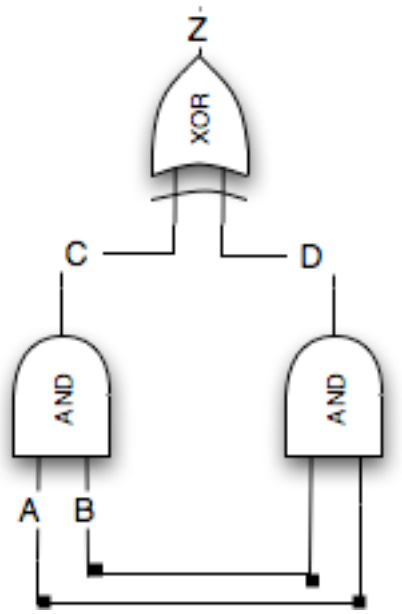
Of course we can solve the SAT instance directly and that would suffice, however, in most cases this will not work because the SAT instances generated by real-world circuits are very hard to solve, therefore, before we invoke a SAT solver, we must simplify our circuit.

We simplify our miter by using structural hashing, deterction of intermediate functional equivalences and circuit re-wiring to first simplify the problem. At each step we produce a proof fragment. Taken together all these proofs fragments can be used to output a valid proof. A problem which we must solve is that we cannot use the intermediate resolution proof of a SAT solver as an overall proof of the entire circuit, instead we present a lifting technique to solve this problem.

Most DPLL solvers do not directly henerate an UNSAT proof, however, Zhang and Malik showed how a modern DPLL SAT solver can be modified to output a valid proof. We us e Minisat 1.41 which already implement a proof logging feature. As an example consider the following two simple circuits:



We build a miter for them as follows:



We can interpret this as a CNF formular:

$1.(\overline{C} \vee A)$

$2.(\overline{C} \vee B)$

$3.(C \vee \overline{A} \vee \overline{B})$

$4.(\overline{D} \vee A)$

$5.(D \vee B)$

$6.(D \vee \overline{A} \vee \overline{B})$

$7.(C \vee D \vee \overline{Z})$

$8.(C \vee \overline{D} \vee Z)$

$9.(\overline{C} \vee D \vee Z)$

$10.(\overline{C} \vee \overline{D} \vee \overline{Z})$

$11.(Z)$

Where the first 3 clauses are for the first AND gate, the second three clauses are for the second AND gate, the other 4 clauses are for the XOR gate and the last clauses, is the output.
If we pass this directly to a SAT solver, it will output UNSAT and produce the following proof by resolution:

$1p.(C \vee \overline{D} \vee \overline{B})[from-3\&4]$

$2p.(C \vee \overline{D})[from-5\&1p]$

$3p.(\overline{C} \vee D \vee \overline{A})[from-2\&6]$

$4p.(\overline{C} \vee D)[from-1\&3p]$

$5p.(\overline{C} \vee \overline{D})[from-10\&11]$

$6p.(C \vee D)[from-7\&11]$

$7p.(\overline{D})[from-2p\&4p]$

$8p.(D)[from-4p\&6p]$

$9p.()$

Which essentially is the proof that the two circuits are equivalent.

In this paper we improve this approach to be faster and more efficient.