Equivalence Checking By Logic Relaxation

Eugene Goldberg

FMCAD,

Mountain View, CA, USA

October 3-6, 2016

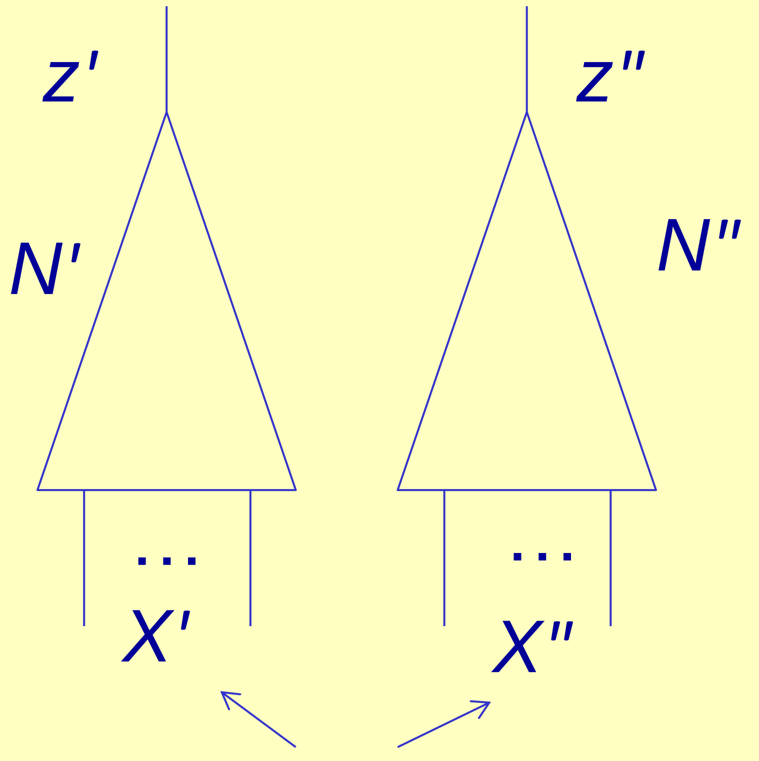
Outline

- Introduction
- Equivalence checking by logic relaxation
- Experimental results and conclusions

Motivation

- Equivalence Checking (**EC**) is an important part of formal verification
- Any progress in EC empowers logic synthesis
- Short EC proofs for structurally similar circuits
- Ideas of EC of combinational circuits can be re-used in EC of sequential circuits and software

Solving EC



$EQ(X',X'')$

$EQ(\mathbf{x}',\mathbf{x}'') = 1$, iff $\mathbf{x}' = \mathbf{x}''$

Prove

$EQ \wedge G_{\mathrm{rlx}} \Rightarrow (z' \equiv z'')$,

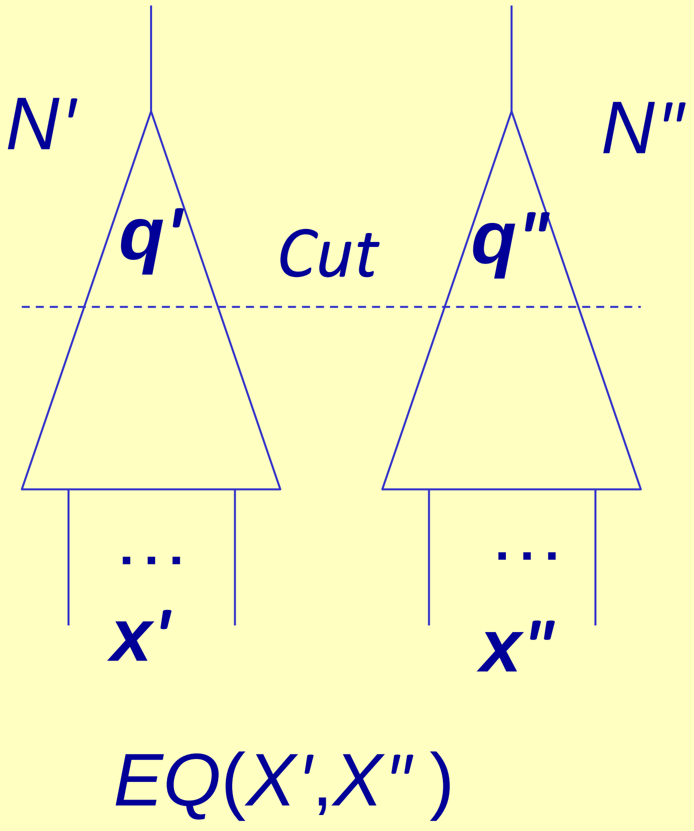
where $G_{\mathrm{rlx}} = F_{N'} \wedge F_{N''}$

This reduces to proving

$EQ \wedge G_{\mathrm{rlx}} \wedge \sim(z' \equiv z'')$

UNSAT

Cut Image



Let Img_{cut} specify the cut image

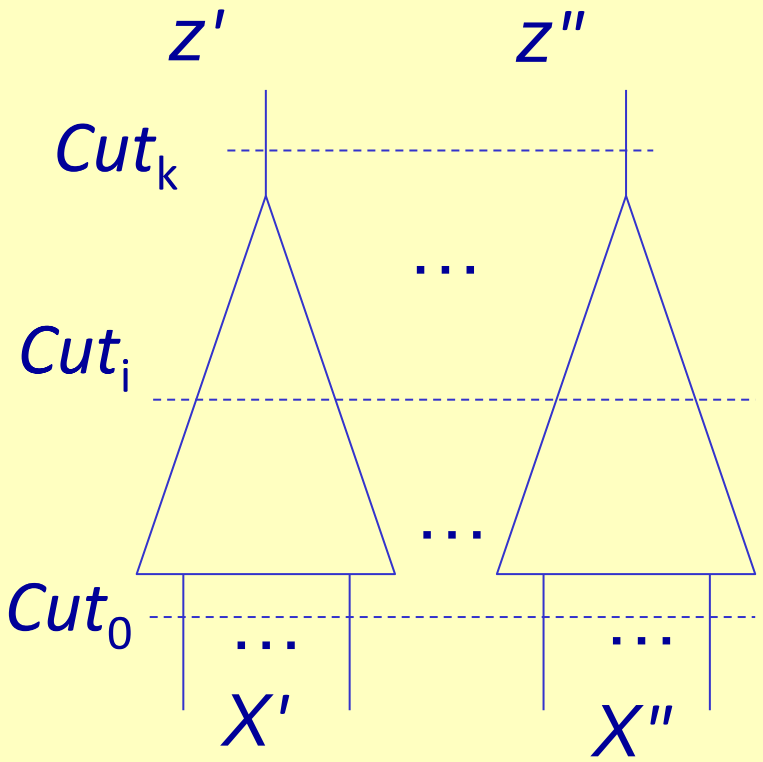
$Img_{cut}(\boldsymbol{q'},\boldsymbol{q''})=0$, iff there is no input $(\boldsymbol{x'},\boldsymbol{x''})$, $\boldsymbol{x'} = \boldsymbol{x''}$ for which N',N'' produce $(\boldsymbol{q'},\boldsymbol{q''})$

Let $Cut = \{z',z''\}$.

N' and N'' are equivalent iff

$Img_{cut} \Rightarrow (z' \equiv z'')$,

Problem To Solve: Finding an Inductive Proof Of Equivalence



Given combin. circuits N' and N'', find formulas H_i such that

- $Img_i \Rightarrow H_i$, $0 \le i < k$
- H_i are as simple as possible
- H_i can be derived from H_{i-1}
- $H_k \equiv Img_k(z',z'')$

A simple inductive proof should exist if N' and N'' are struct. similar

Some Background

Building inductive proofs of equivalence

- Berman, Trevillyan 1988
- Brand 1993
- Kuehlmann, Krohm 1996
- Goldberg, Prasad, Brayton 2001
- Mishchenko,Chatterjee,Brayton,Een 2006

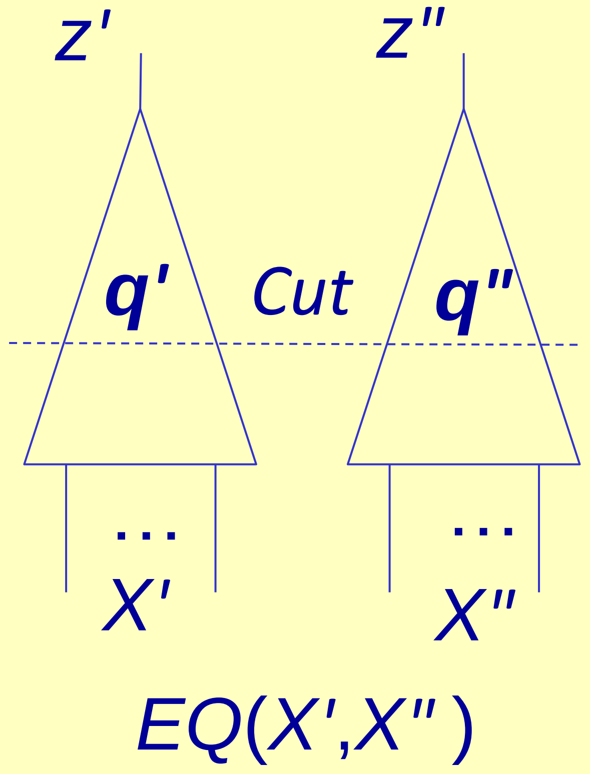
Proofs are based on derivation of pre-defined relations e.g. equivalences

Outline

- Introduction
- Equivalence checking by logic relaxation
- Experimental results and conclusions

Structure Of Cut Image

Assignments excluded from cut image: $S_{excl} = S_{rlx} \cup S_{imp}$



S_{rlx} = {(***q'***,***q"***) | only relaxed inputs (***x'***,***x"***) where ***x'*** ≠ ***x"*** can produce (***q'***,***q"***) }

S_{imp} = {(***q'***,***q"***) | no input (***x'***,***x"***) can produce (***q'***,***q"***) }

(***q'***,***q"***) $\in S_{imp}$ iff

- ***q'*** cannot be produced in *N'* and/or
- ***q"*** cannot be produced in *N"*

Definition Of Boundary Formulas

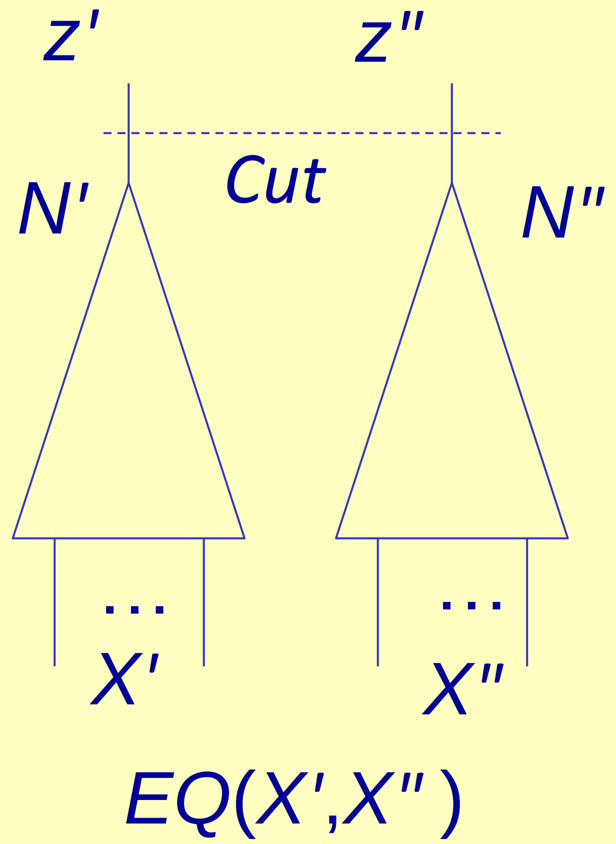
EC by Logic Relaxation:

"replace" Img_{cut} with boundary formula H_{cut}

Boundary formula H_{cut} :

1. If $(\boldsymbol{q}',\boldsymbol{q}'') \in S_{\text{rlx}}$, then $H_{\text{cut}}(\boldsymbol{q}',\boldsymbol{q}'') = 0$
2. If $(\boldsymbol{q}',\boldsymbol{q}'') \in S_{\text{imp}}$, then $H_{\text{cut}}(\boldsymbol{q}',\boldsymbol{q}'')$ can take an arbitrary value
3. $Img_{\text{cut}} \Rightarrow H_{\text{cut}}$

Boundary Formula for *Cut* = {*z'*,*z''* }



Assume that *N'* and *N''* are not constants

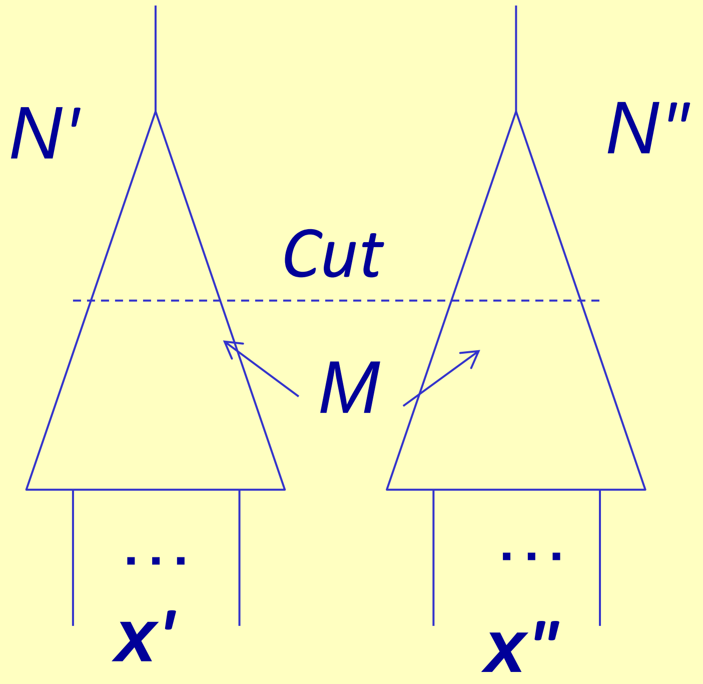
$$S_{imp} = \varnothing \Rightarrow S_{excl} = S_{rlx}$$

$$\Downarrow$$

$$H_{cut} \equiv Img_{cut}$$

Testing if *N'* is a constant: two easy SAT checks

Boundary Formula And Partial Quantifier Elimination



$EQ(X',X'')$

Complete Quantif. Elimin.

$Img_{cut} \equiv \exists W\,[\,EQ \wedge F_M]$

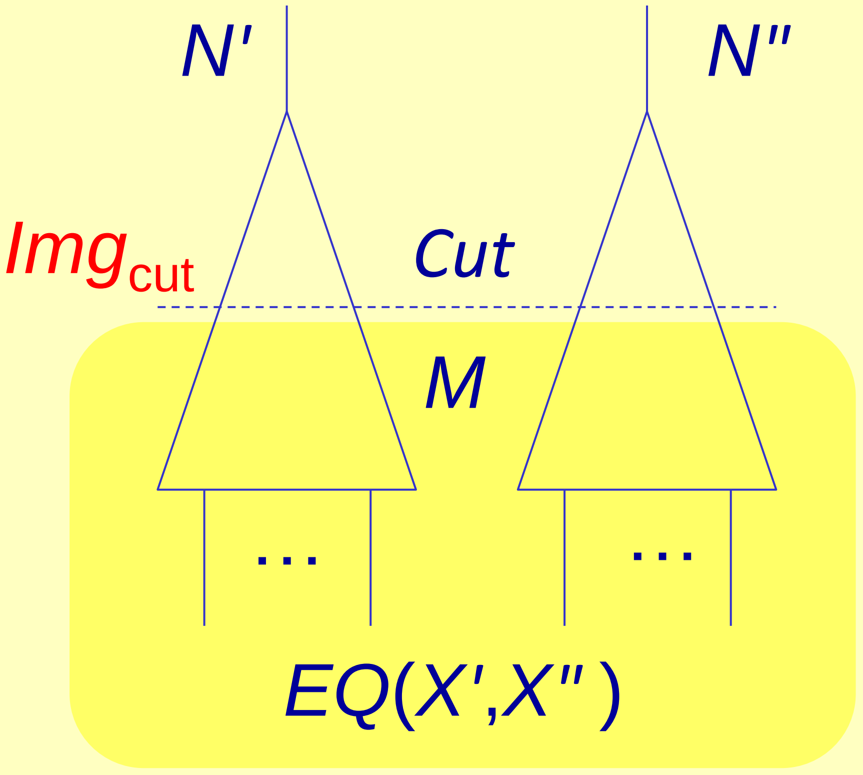
$W = Vars(F_M) \setminus Vars(Cut)$

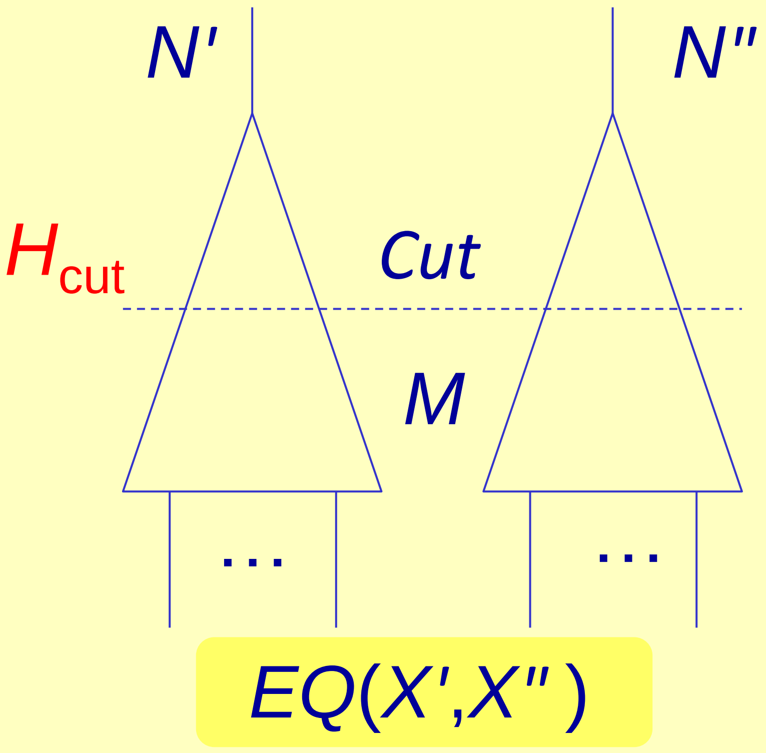
Partial Quantif. Elimin.

$H_{cut} \wedge \exists W\,[\,F_M] \equiv \exists W\,[\,EQ \wedge F_M]$

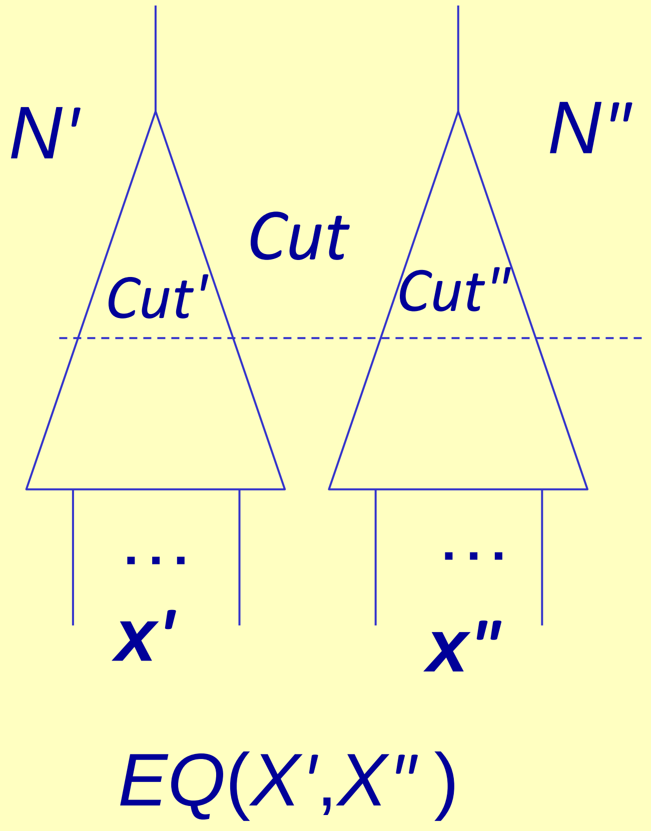
$EQ \wedge G_{rlx} \wedge \sim(z' \equiv z'')$ is equisat. with
$H_{cut} \wedge G_{rlx} \wedge \sim(z' \equiv z'')$
where $G_{rlx} = F_{N'} \wedge F_{N''}$

Contrasting Cut Image And Boundary Formulas





Boundary Formulas Of Structurally Similar Circuits



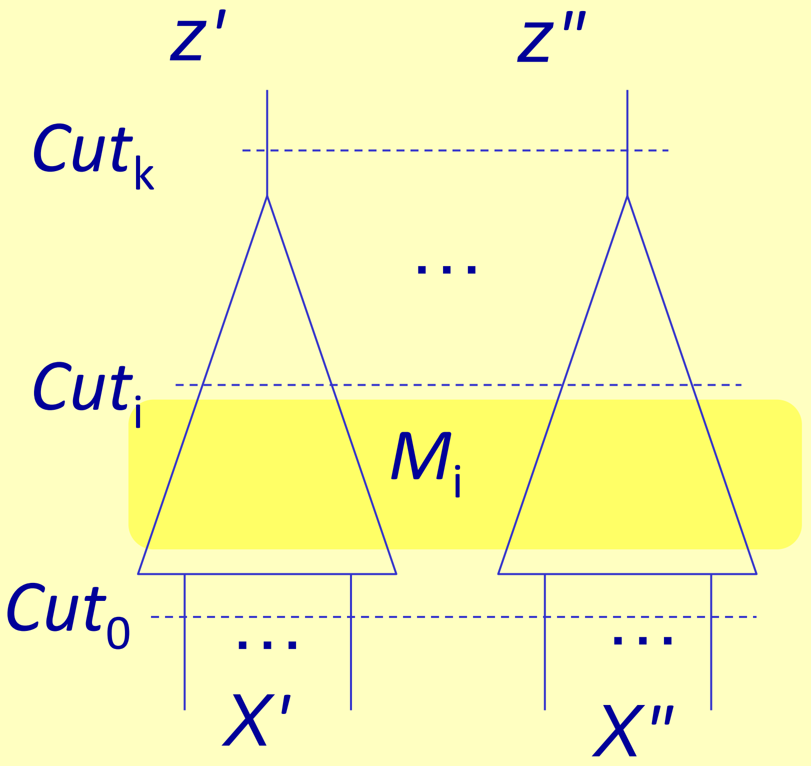
Suppose, $\forall v \in Cut'$

$v = g_v(L_v)$ where $L_v \subseteq Cut''$

Let Max_{cut} be the largest value of $|L_v|$, $\forall v \in Cut'$

Then H_{cut} can be built from $(Max_{cut} + 1)$-literal clauses

EC By Logic Relaxation



$Cut_0 = X' \cup X'', \ldots, Cut_k = \{z', z''\}$

Compute $H_0, .., H_k$

where $H_0 = EQ(X', X'')$

$H_i \wedge \exists W_i\, [\, F_{Mi}\,] \equiv \exists W_i\, [H_{i-1} \wedge F_{Mi}]$

$W_i = Vars(F_{Mi}) \setminus Vars(Cut_i)$

If $H_k \Rightarrow (z' \equiv z'')$,

N' and N'' are equivalent

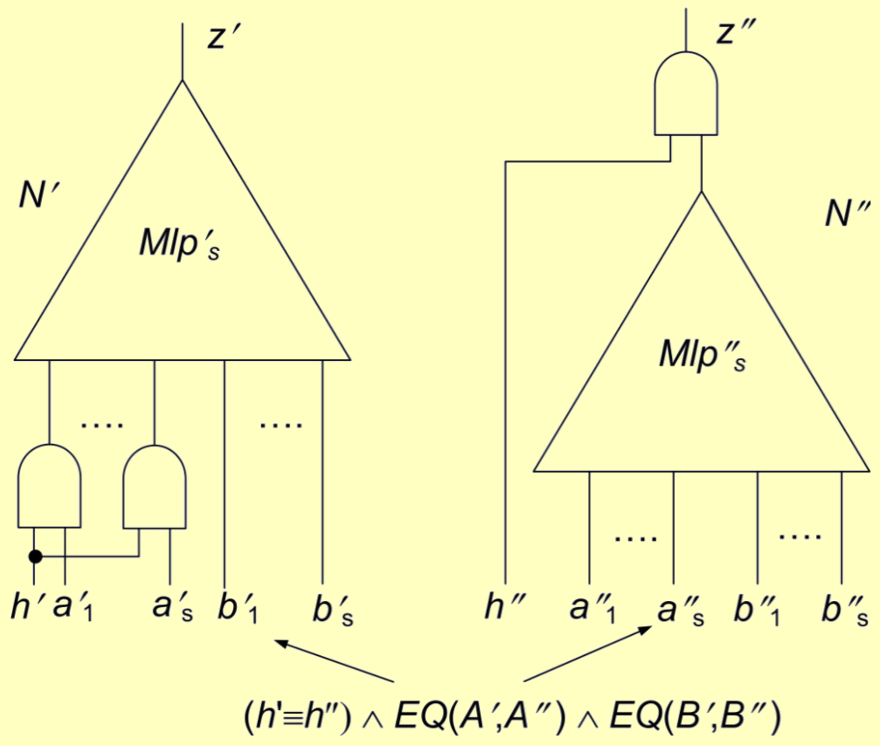
If, say, $H_k(z' = 0, z'' = 1) = 1$ and N', N''

can produce 0 and 1, they are inequivalent

Outline

- Introduction
- Equivalence checking by logic relaxation
- Experimental results and conclusions

Non-Trivial Example Of EC



z'
z''
N'
N''
Mlp'_s
Mlp''_s
....
h' a'_1 a'_s b'_1 b'_s
h'' a''_1 a''_s b''_1 b''_s
$(h' \equiv h'') \wedge EQ(A', A'') \wedge EQ(B', B'')$

Mlp_s computes a median bit of an s-bit multiplier

Operands A and B where $A=\{a_1,..,a_s\}$, $B=\{b_1,...,b_s\}$

h is an additional input variable

If h=1, then N' and N'' compute Mlp_s

if h=0, then N' and N'' evaluate to 0

Comparison With *ABC*

- Partial Quantifier Elimination (a variation of HVC-14 algorithm) is based on machinery of D-sequents (FMCAD-12 , FMCAD-13)
- *ABC* is a high-quality tool developed at UC, Berkeley

val. of s in *Mlp*$_s$	**#cuts**	**EC by LoR (s.)**	***ABC* (s.)**
10	37	**4.5**	10
11	41	**7.1**	38
12	45	**11**	142
13	49	**16**	757
14	53	**25**	3,667
15	57	**40**	11,237
16	61	**70**	> 6 h

Formulas H_i were computed approximately

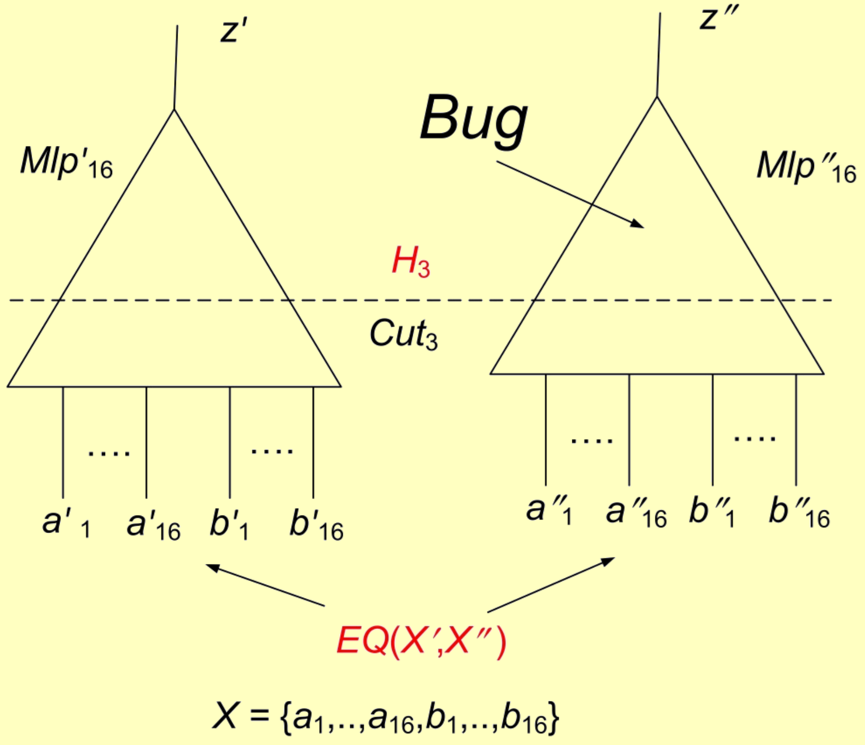
$$\Downarrow$$

$$H_i \wedge \exists W_i\, [\, F_{Mi}\,] \equiv \exists W_i\, [H_{i-1} \wedge F_{Mi}]$$

F_{Mi} specifies logic below *i*-th cut

Only a subset of clauses of F_{Mi} was used

Proving Inequivalence



Formula α

$$EQ(X',X'') \wedge F_{N'} \wedge F_{N''} \wedge \sim(z' \equiv z'')$$

Formula β

$$H_3 \wedge F_{N'} \wedge F_{N''} \wedge \sim(z' \equiv z'')$$

Formula H_3 was computed **precisely**

Sat-solver : Minisat 2.0, *Time limit*: 600 s

Form. type	**#solved**	**total time (s)**	**median time (s)**
α	95	> 3,490	4.2
β	**100**	**1,030**	**1.0**

Conclusions

- Relative_complexity(N',N'') << Absolute_complexity(N',N'')
- EC by logic relaxation gives a general solution
- It can be extended to sequential circuits/programs
- Efficient partial quantifier elimination is of great value