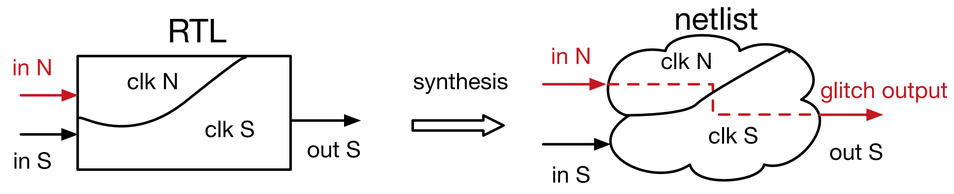
Defining and Detecting Synthesis-Generated Glitches

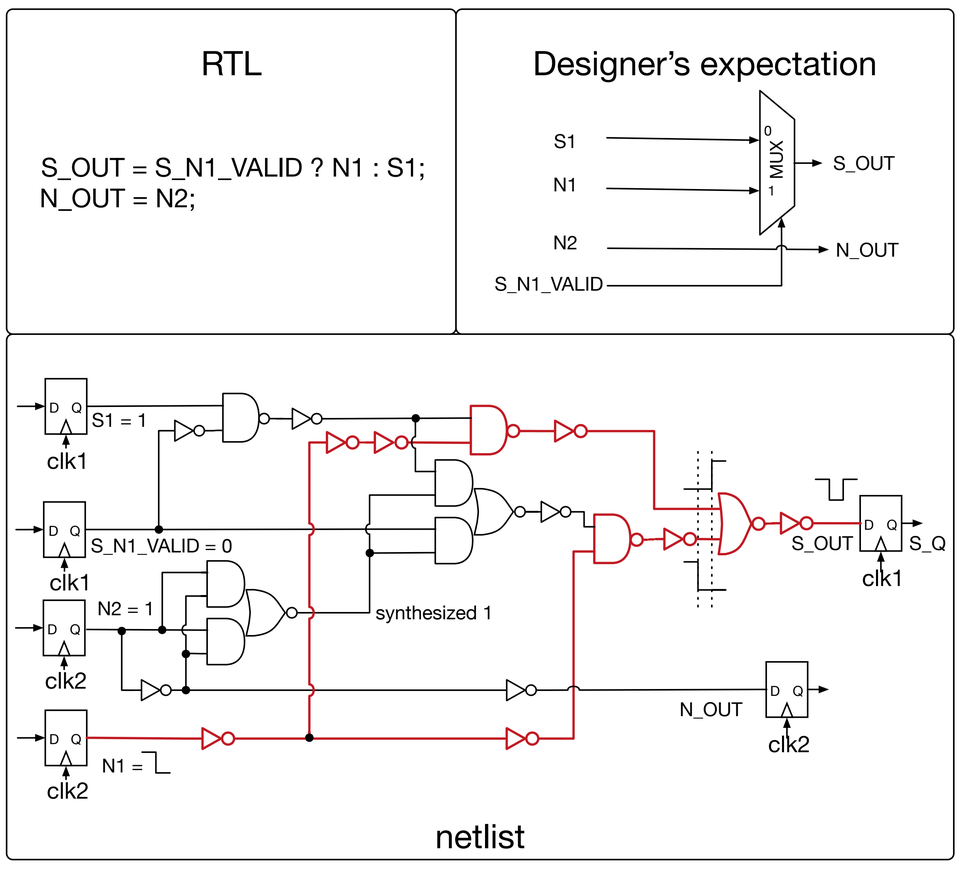
Yan Peng yanpeng@cs.ubc.ca
Mark R. Greenstreet mrg@cs.ubc.ca
Ian W. Jones ian.w.jones@oracle.com

Motivation



Glitch: a transition on a non-synchronous signal can cause the output of the combinational logic to temporarily change to an unstable value.
Synthesis-generated Glitch: synthesis tools can introduce glitches. This can happen even though the RTL design is free of such a glitch.

Problem: An example



- The netlist is boolean-logically equivalent to the RTL and passes standard logic equivalence checks
- A glitch can propagate through the netlist, while the designer intended the RTL to block such a glitch
- Synthesis can refactor a mux to shorten a critical path
- Synthesis can add extra gates to satisfy min-hold-time constraints, especially for designs with high clock frequencies

Formalization

For each combinational logic fan-in tree (DAG) with output q: $G_{q,\mathsf{RTL}}(s_q, n_q)$ denotes the value of q according to the RTL, and $G_{q,\mathsf{net}}(s_q, n_q)$ denotes the value of q according to the netlist. To detect glitches, we look for an assignments to s_q and n_q for which $G_{q,\mathsf{net}}$ propagates an $\mathbf{X}$ to its output, but $G_{q,\mathsf{RTL}}$ does not. More formally,

$$\begin{aligned}\text{glitchFree}(q) \;=\; &\forall s_q \in \mathbb{B}^{|s_q|}.\ \forall n_q \in \mathbb{B}^{\mathbf{X}^{|n_q|}}.\\ &(G_{q,\mathsf{net}}(s_q, n_q) = \mathbf{X}) \Rightarrow (G_{q,\mathsf{RTL}}(s_q, n_q) = \mathbf{X})\,.\end{aligned} \tag{1}$$

This says that the netlist produces no "unexplained" $\mathbf{X}$ values. Checking property glitchFree is complete for co-NP. [a]
A stricter notion of a netlist being glitch-free:

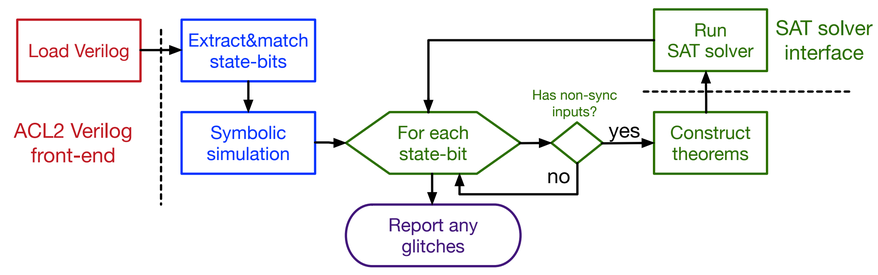
$$\begin{aligned}\text{glitchFree}_2(q) \;=\; &\forall s_q \in \mathbb{B}^{|s_q|}.\ \exists b \in \mathbb{B}.\\ &(\forall n_q \in \mathbb{B}^{|n_q|}.G_{q,\mathsf{net}}(s_q, n_q) = b)\\ \Rightarrow\quad &G_{q,\mathsf{net}}(s_q, X^{|n_q|}) = b\,.\end{aligned} \tag{2}$$

This says that for every valuation of the synchronous inputs for which the output does not depend on boolean-valued, non-synchronous inputs, $\mathbf{X}$ values for the non-synchronous inputs do not affect the output either. Property glitchFree_2 does not require an RTL description of the function; however, checking glitchFree_2 is complete for Π_2. [b]

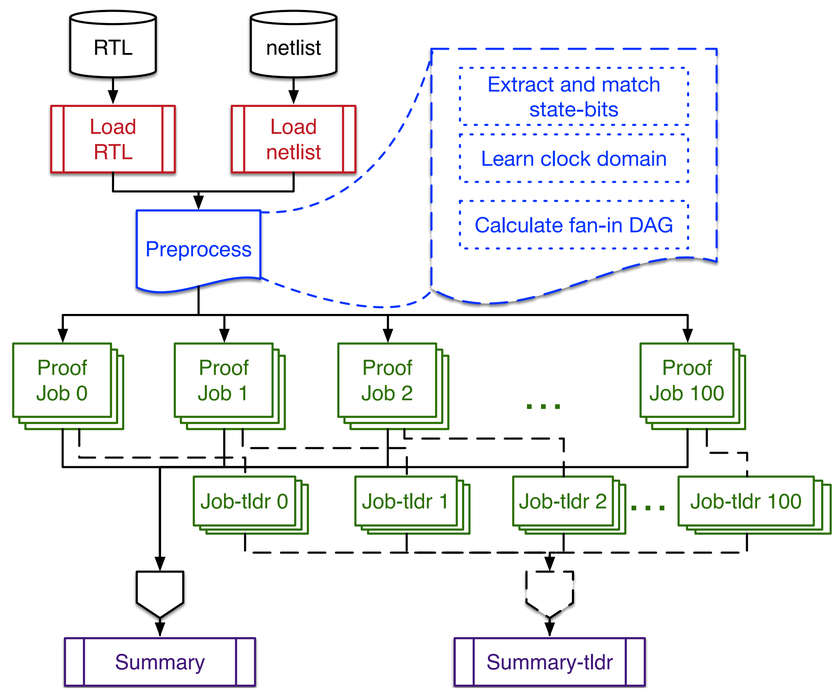
[a] A property is *in* co-NP if it can be written as $\forall x.\, P(x)$ where x is a boolean vector.
[b] A problem is in Π_2 if it can be written as $\forall x.\, \exists y.\, P(x, y)$. Π_2 is "harder" than NP or co-NP in that any problem in NP or co-NP is also in Π_2.

Glitch Hunter

- **Sequential Glitch Hunter**: ACL2 provides a comprehensive Verilog front end and a SAT solver interface. Theorems are automatically generated.

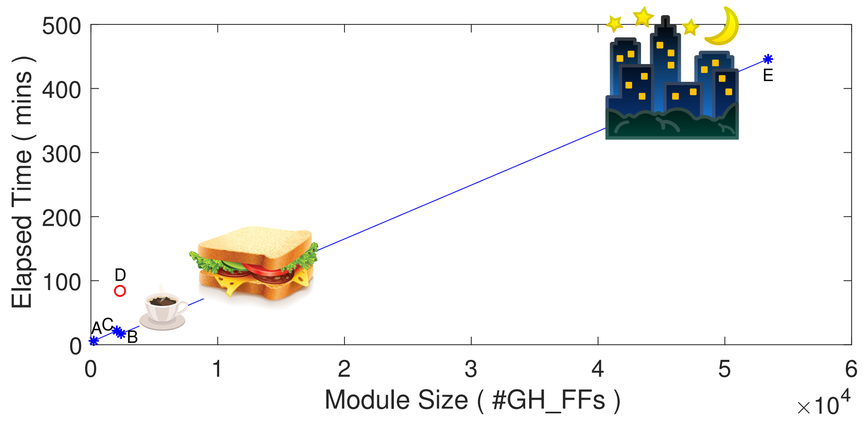


- **Parallel Glitch Hunter**: distribute computation over multiple machines by leveraging the ACL2 certification method and the Unix Make utility.



Results

Module	#gates	#FFs	#GH-FFs[a]	T_{32}[b]
A	1264	721	221(30.7%)	6
B	10923	4256	2378(55.9%)	17
C	90432	14874	2045(13.7%)	22
D	29018	5092	2293(45.0%)	84
E	238783	177996	53415(30.0%)	446



1. For modules with a few thousand gates, the time to dispatch jobs in the cluster dominates, and 16 or 32 processors seems optimal
2. For modules with hundreds of thousands of gates, the preprocessing step becomes a sequential bottleneck accounting for about 2% of the total computation and limiting speed-up to around 50
3. Outlier module D for preprocessing time due to combinational loops

[a] GH-FFs are state-bits that include non-synchronous inputs in their fan-in trees
[b] T_{32} is the time (in minutes) with 32 parallel jobs

Conclusion & Future Work

We presented a precise, logical definition of synthesis-generated glitches. We implemented a tool that solves this problem for real modules from designs in industry. We demonstrated a parallel implementation that runs on linux server clusters that are standard in industry. We hope commercial CDC verification tools can benefit from this approach.