Specialized Strategies for Learning Integrated Circuits using Angluin L* and Rivest/Shapire Homing Inference

Tanya Braun, Arne Wichmann, and Sibylle Schupp

Institute for Software Systems, Hamburg University of Technology, Hamburg, Germany
{tanya.braun,arne.wichmann,schupp}@tuhh.de

In a perfect world, every digital integrated circuit (IC) leads a well-documented life. Alas, documentations are lost, not written, deliberately withheld. To reconstruct the inner workings of an IC, we modify the learning algorithms by Dana Angluin and by Ronald Rivest and Robert Schapire so that they employ ICs as the learning environment. Known pin directions and functions allow reducing the state space, alphabet size, and set of learners and therefore accelerate the learning. We set up different strategies to seek out hidden state to realize an approximative equivalence check and provide the necessary counterexamples. In summary, the contributions are different strategies for the teacher to test for equivalence and a specialized learner for ICs, called `ALICe` [3], with the possibility to incorporate prior knowledge.

Angluins L* algorithm [1] learns an automaton representation using the inputs and the corresponding results plus the counterexamples obtained from a teacher, assuming we know a reset. Rivest and Schapires algorithm [6] can be seen as a generalization of L* and handles absent resets using homing sequences. Angluin has one learner whose queries are simulated from a unique state. To execute such a query in any environment, the environment must be reset to the unique state. In the absence of a reset, homing sequences bring the environment into a defined state. The learner, Angluins main learning loop, becomes the basic building block. Rivest and Schapires algorithm maintains a set of learners to accommodate the different states a homing sequence may lead to. For each final state the homing sequence can lead to, a learner exists that has this final state as a starting state.

`ALICe` uses the `libalf` library for the learners [2] and builds the homing algorithm around it. The IC takes the role of the teacher in the Angluin learning. `ALICe` incorporates prior knowledge about the ICs interface (see Table 1) into the model of the input alphabet and the learning process.

Prior Knowledge	Usage	Effect
Pin directions	Input = in + inout Output = out + inout	Reduced number of learners/ states, reduced alphabet size
Clock pin	Subtract from input alphabet, let hardware handle clock behaviour	Reduced alphabet size, reduced state space
Clear word	As homing sequence	One learner
Clear pin	Build homing sequence, subtract from input alphabet	One learner, reduced alphabet size

Table 1: Usage and Effects of Prior Knowledge

1 Strategies to Realize Equivalence Checks

The teacher has to provide a counterexample to an incorrect representation of an IC. An ideal check is impossible due to the infinite amount of stimulation necessary. We therefore

	Strategy	Expected Effect
0	No check, without any knowledge	Fewer queries than with any strategy, missing states with clocked ICs
0c	No check, subtract clock pin	Fewer queries, fewer missing states with clocked ICs
0d	No check, subtract clock and clear pin	Fewer queries than 0c, same state counts
I	Walk to each state, check two times the input symbols; **base case**	Rather high number of learners, states, queries
Ia	Base case strategy, clear word for reset (construct from pin, all zeros)	Only one learner (for those with a correct reset)
Ib	Base case strategy, clear pin for reset, subtract it from input alphabet	Only one learner, fewer queries due to reduced alphabet
Ic	Base case strategy, subtract clock pin (let teacher handle clock behaviour)	Reduced state space, fewer queries due to reduced alphabet/state space
Id	Base case strategy, clear pin for reset, subtract clear and clock pin	Only one learner, reduced state space, fewer queries
II	Walk to each state and toggle each pin	Fewer queries than with I, missing states
III	Walk to each state; for each input, stimulate and toggle each pin	More queries and states than with II, performance similar to I
IV	Walk to each state, stimulate ten random inputs, then toggle each pin	Fewer queries than with base case, more states than with II
V	Block other learners and test with random inputs	Fewer queries than with base case

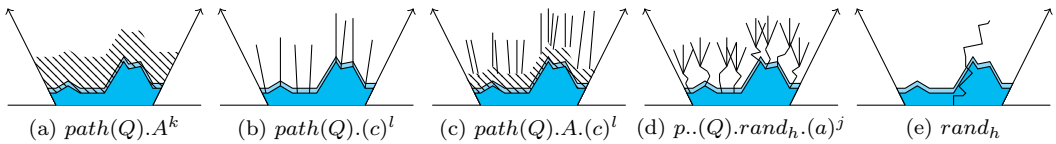
Table 2: Test Strategies for Evaluation

introduce different strategies, which are based on a general approximate equivalence check $path(Q)?.A^k.rand_h.[(a)^j|(c)^l]$, and list their expected effects (see Table 2).

The strategies **I** to **IV** use a $path(Q)$ option to walk to the learned states and perform additional stimulation from the fringe of the explored state space. This part of the strategies avoids repeating questions that were already posed by the learner. In Fig. 1 these parts of the strategies are visualised as the coloured area.

The **0** strategies do not perform an equivalence check. Strategy **I** appends all possible combinations of two input alphabet elements and densely explores the area next to the learned area (see Fig. 1a). Strategy **II** tries to find first counterexamples fast using a toggle check plus path option (see Fig. 1b). A hybrid strategy (**III**) appends each symbol of the input alphabet and then toggles all pins (see Fig. 1c). The last path-based strategy (**IV**) appends 10 random symbols and then toggles all pins (see Fig. 1d).

A conceptually different strategy (**V**) does not systematically explore the state space, but instead performs a random walk using the input symbols to find a counterexample (see Fig. 1e). This strategy is close to the weak oracle introduced by Angluin and Rivest and Schapire.



(a) $path(Q).A^k$ (b) $path(Q).(c)^l$ (c) $path(Q).A.(c)^l$ (d) $p..(Q).rand_h.(a)^j$ (e) $rand_h$

Figure 1: The Search Space Covered by Equivalence Strategies[1]

[1] The x-axis represents the inputs as queries. The y-axis indicates the query length. The dark blue area represent the queries that lead to new states. The light blue area constitutes the input the learner looked further.

2 Evaluation

We evaluated the different equivalence strategies and applied levels of prior knowledge to several groups of ICs according to the number of queries, number of learners, and various performance parameters.

We use groups of ICs that stem from three university introductory courses to digital circuits [7, 4, 5]. The stateless groups include combinatoric logic gates like NAND with different numbers of input pins, de/encoders, (de)multiplexers, and arithmetic entities like comparators or full adders. The groups of ICs with state include shift registers, transceivers and buffers, flip-flops with clear pins or inverted outputs, and counters and oscillators. The evaluation shows that it is possible to learn the functionality of digital ICs.

Although `ALICe` needs no information about an IC except the number of pins as well as the power supply and ground pin, we can accelerate learning by nearly 100% by providing additional knowledge about pin directions and functions for selected groups.

Different equivalence checks vary considerably in their number of queries. The random-based strategies, often found in the literature, usually trigger a reset of the IC and terminate learning before an useful result is obtained (**IV**, **V**). On the other hand, for stateless ICs, an equivalence check is not necessary, and a **0** strategy with minimal cost is sufficient.

References

[1] Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.

[2] Benedikt Bollig, Joost-Pieter Katoen, Carsten Kern, Martin Leucker, Daniel Neider, and David R. Piegdon. libalf: The automata learning framework. In *Proceedings of the Twentysecond International Conference on Computer Aided Verification*, pages 360–364, 2010.

[3] Tanya Braun. Teaching angluin to learn the inner workings of integrated circuits. Master's thesis, Hamburg University of Technology, Hamburg, Germany, August 2015.

[4] Anantha Chandrakasan. 6.374: Analysis and design of digital integrated circuits. Massachusetts Institute of Technology: MIT OpenCourseWare. Lecture Notes, 2003.

[5] Anantha Chandrakasan. 6.111: Introductory digital systems laboratory. Massachusetts Institute of Technology: MIT OpenCourseWare. Lecture Notes, 2006.

[6] Ronald L. Rivest and Robert E. Schapire. Inference of finite automata using homing sequences. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, pages 411–420, 1989.

[7] Sven-Ole Voigt. Technische Informatik: Skript zur Vorlesung. Hamburg University of Technology. Lecture Notes, 2013.