Handshake Verification in WORKCRAFT

Victor Khomenko†, Danil Sokolov†, Alex Yakovlev†, David Lloyd‡
{victor.khomenko, danil.sokolov, alex.yakovlev}@ncl.ac.uk; david.lloyd@diasemi.com
†Newcastle University, UK; ‡Dialog Semiconductor, UK

I. WHY HANDSHAKE VERIFICATION?

Many AMS applications, such as PMIC, require low-latency "little digital" controllers [1]. In order to meet the latency requirements in a traditional synchronous controller, semiconductor industry has to push high-frequency clocking to the limits, which is too expensive. An alternative is to use asynchronous design that is supported by WORKCRAFT software (https://workcraft.org/). Predictability of the design flow and provable correctness of the obtained circuits make WORKCRAFT particularly attractive for industry [1], [2].

Handshakes (h/s) are a common replacement for the global clock and thus are fundamental in asynchronous design. Even though the h/s protocol seems trivial on the surface, it does have a number of pitfalls and so must be formally verified: In our experience, students and engineers do occasionally get it wrong. This paper demonstrates how h/s verification of STGs [4], [3] has been implemented in WORKCRAFT. We focus on control h/s (and so set aside the issues of data validity) as STGs are control-oriented.

For example, consider a DECOUPLER: It communicates with two modules, LEFT and RIGHT, by the h/s rl / al and rr / ar, respectively. The idea is that LEFT can quickly complete its h/s with DECOUPLER and continue its execution, while DECOUPLER takes care of completing the h/s with RIGHT.

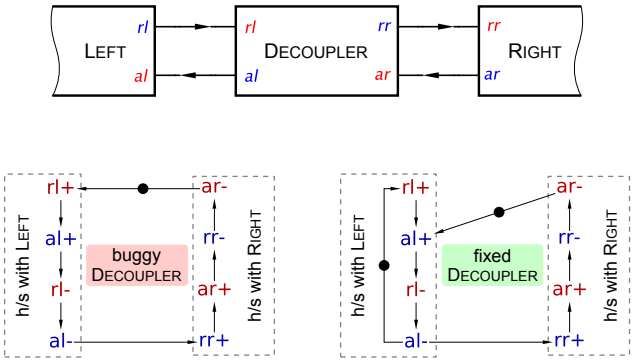

Figure 1. DECOUPLER module.

Consider the (buggy) STG specification of DECOUPLER on the left. Setting aside the issues of data validity and efficiency (one can remove the negative signal edges from the critical path) there is a serious problem in this specification: LEFT can attempt to send another request before DECOUPLER completes its h/s with RIGHT – indeed, LEFT has no way of knowing whether that h/s is completed. However, the specification does not capture this possibility, and assumes that LEFT will wait for the h/s between Decoupler and Right to complete. In other words, the assumptions about the environment's behaviour are incorrect: To fix this, one must enable rl+ immediately after al-, see the STG on the right.

The above bug can be caught if one verifies the N-way conformation property of the overall system: indeed, LEFT can send an unexpected input to DECOUPLER, so the N-way conformation is violated. However, it would be much better to catch such bugs at the level of a module, making the verification much more efficient, and also helping with debugging – a violation trace for the N-way conformation will likely include some activity in other modules, which is largely irrelevant but can make the trace very long.

II. HANDSHAKE ASPECTS

This section gives a brief overview of various h/s aspects, which make their verification non-trivial.

Multi-signal requests and acknowledgements: It is possible (in fact, common) that more than two signals participate in a h/s. For example, there can be a dual-rail (or, in general, multi-rail 1-hot) request (e.g. to specify the mode of operation) with a single-rail acknowledgement. Similarly, the acknowledgement can be dual-rail (or, in general, multi-rail 1-hot) to return some information to the caller. Hence, we assume that requests and acknowledgements are two non-empty sets of signals of the same type (either input or output), and the type of requests is opposite to the type of acknowledgements. The h/s is called *active* if the requests are outputs (i.e. the module initiates the h/s) and *passive* if the requests are inputs (i.e. the environment initiates the h/s). Moreover, at most one request is allowed to be asserted at any time, and similarly for acknowledgements (this is a part of verification).

Signal order: For a h/s r / a, suppose the initial values of r and a are 0. The h/s protocol requires that these signals follow the order r+ a+ r- a-... That is, there are four different states in a h/s, uniquely determined by the values of signals r and a, with the following requirements:

r=0 & a=0: r-, a+, a- must be disabled;
r=1 & a=0: r+, r-, a- must be disabled;
r=1 & a=1: r+, a+, a- must be disabled;
r=0 & a=1: r+, r-, a+ must be disabled.

These conditions can be generalised to h/s comprising multiple requests and/or acknowledgements in a natural way, as at most one request and at most one acknowledgement can be asserted at any time. Note that these properties only require certain signal edges to be disabled. The receptiveness property discussed below imposes some enabledness requirements.

Receptiveness: The pitfall in the DECOUPLER example illustrates an important receptiveness property that should normally hold for h/s (there are some exceptions due to dependencies between different h/s, such as a choice or sequencing). Suppose there is a single request r and a single acknowledgement a, and their initial values are 0. Then for a passive h/s:

r=0 & a=0: r+ must be enabled;

r=1 & a=1: r- must be enabled;

and for an active h/s:

r=1 & a=0: a+ must be enabled;

r=0 & a=1: a- must be enabled.

These conditions can be generalised to h/s comprising multiple requests and/or acknowledgements, as at most one request and at most one acknowledgement can be asserted at any time.

Initial state of handshake: In some situations it may be convenient to initialise the circuit in a state that is different from the conventional initial state with both request and acknowledgement withdrawn, e.g. to remove the initialisation logic from a critical path. However, one has to be very careful when doing so, as the h/s may progress further than intended during the initialisation.

Signal inversions: One can often optimise the circuit implementation by changing the polarity of some signals – this may allow one to remove some inverters and/or replace positive logic gates by negative ones.

III. WORKCRAFT HANDSHAKE WIZARD

WORKCRAFT h/s wizard is shown in Fig. 2 (the dialog appearance depends on whether the h/s is passive or active). It allows the user to select the signals and aspects of the h/s and then automatically formulates the necessary h/s properties to be formally verified. If the h/s protocol is violated, a violation trace is reported and can be simulated in WORKCRAFT. For example, for the buggy DECOUPLER the trace rl+ al+ rl- al- is reported: After this trace the receptiveness is violated as rl+ is expected to be enabled, but it is not.

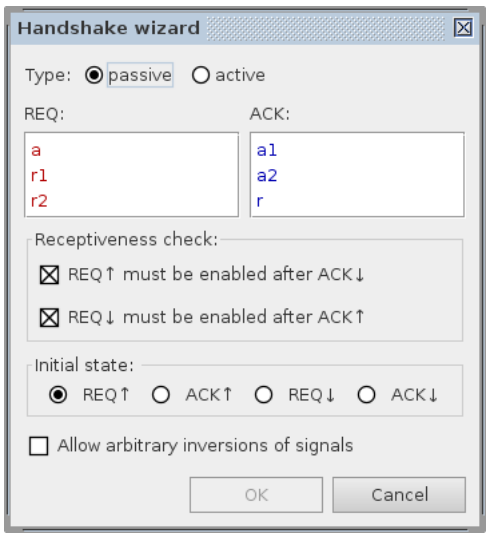

Figure 2. Handshake wizard.

Consider the STG in Fig. 3. The passive h/s r1 / a1 and r2 / a2 are mutually exclusive, as r1+ and r2+ disable each other. Hence, e.g., in a state with r1=0 & a1=0, edge r1+ is not necessarily enabled as r2+ can fire and disable r1+. In such a situation it would be reasonable to skip the receptiveness checks for r1+ and r2+ (while still keeping them for r1- and r2-), which can be done by unchecking the corresponding checkbox in the *"Receptiveness check"* group. This, however, is not perfect, as certain receptiveness is still required: In this case it would be reasonable to check that r1+ is enabled whenever r1=0 & a1=0 & r2=0 & a2=0. However, it would be infeasible to support all the imaginable dependencies between h/s. Alternatively, since the h/s are mutually exclusive, one can treat them as a single h/s {r1, r2} / {a1, a2}, which would satisfy the receptiveness property. However, this is not perfect either, as one may want to ensure that r1 is acknowledged specifically by a1 rather than {a1, a2}. Hence, one can combine all these checks, i.e. verify h/s r1 / a1 and r2 / a2 with relaxed receptiveness, as well as {r1, r2} / {a1, a2}.

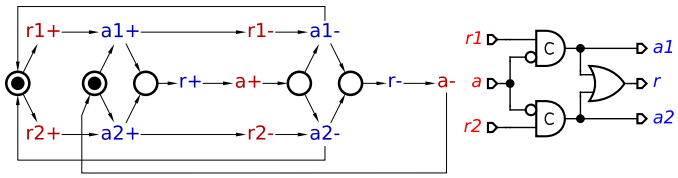

Figure 3. An STG with mutually exclusive h/s and the synthesised circuit.

Note that if both r1 / a1 and r2 / a2 h/s are with the same module, the system could be re-designed by sharing the acknowledgement signal (by collapsing a1 and a2 into one signal a12), resulting in a h/s {r1, r2} / a12 with a dual-rail request and a single-rail acknowledgement. The modified STG passes the receptiveness checks and results in a simpler circuit.

By changing the polarity of a, one can get rid of the two "bubbles" at the inputs of the C-elements in the circuit. Furthermore, by changing the polarity of r one can turn the OR-gate into a NOR-gate. (Of course, these transformations should be reflected in the design of the environment.) To verify h/s a / r in the resulting STG the user must tick the *"Allow arbitrary inversions of signals"* checkbox. (The polarities can be deduced automatically, from the initial state of the h/s and the initial values of the signals participating in the h/s, so this checkbox is just a safety feature to inform the tool that the polarity changes are intentional rather than accidental.)

Acknowledgements: This research was partially supported by Dialog Semiconductor grant *Asynchronous-Analogue Electronics Co-design (A×A)*.

REFERENCES

[1] D. Sokolov et al: *"Automating the Design of Asynchronous Logic Control for AMS Electronics"*, IEEE TCAD, 2019.

[2] D. Sokolov, V. Khomenko, and A. Mokhov, "*Workcraft: Ten years later*," in *This Asynchronous World*, 2016, pp. 269–293.

[3] L. Rosenblum and A. Yakovlev, "*Signal graphs: from self-timed to timed ones*," in *Proc. Int. Workshop on Timed Petri Nets*, 1985, pp. 199–206.

[4] T.-A. Chu: *"Synthesis of self-timed VLSI circuits from graph-theoretic specifications"*, PhD thesis, MIT, 1987.