I. INTRODUCTION

The traditional design methods for speed-independent (SI) circuits [1] require their behaviour to be output-persistent. A common source of non-persistence is arbitration [2] that leads to a choice between output signals. It is the designer’s responsibility to remove such non-persistent behaviour before proceeding to synthesis, usually by manually factoring the arbitration out into the environment, where the choice is implemented using a mutex element [3]. There are several problems with this approach:

- Significant manual effort factoring out the mutex and inserting it after synthesis.
- There is no guarantee that the signals the designer thinks can be implemented by a mutex actually follow the arbitration protocol.
- Factoring out converts mutex grants into inputs, so verification of output-persistence would miss a situation when a mutex grant is disabled due to premature withdrawal of corresponding request (this applies to verification of both, the original specification and the circuit implementation).

In this paper we demonstrate how these problems were solved by integrating automatic mutex insertion into the SI synthesis flow implemented in WORKCRAFT (https://workcraft.org/).

II. DESIGN FLOW

WORKCRAFT takes a circuit specification in form of a Signal Transition Graph (STG) [4]. Choices involving outputs are considered violations of output-persistence by default, so the user must tag choice places meant to be implemented by mutexes. For example, place me in Fig. 1a is tagged as a mutex place (visualised by an outline circle). Otherwise the user designs the STG in a natural way, with mutex grants being output or internal signals (as opposed to them being inputs as prescribed by the traditional factoring-out technique).

By looking at the vicinity of a mutex place the tool can automatically determine the request/grant pairs of the corresponding mutex (let us denote them as r1/g1 and r2/g2). During the output-persistence check, the choice between g1 and g2 (as in Fig. 1a) is not considered to be a violation. However, disabling of a grant by premature withdrawal of the corresponding request (as in Fig. 1b) will still be caught.

The tool then needs to verify that these signals follow the arbitration protocol and the grants g1, g2 can indeed be implemented by a mutex with requests r1, r2. Both, “late release” (grants are mutually exclusive) and “early release” (a grant can be issued before the other grant is reset, as long as its request is withdrawn) versions of the arbitration protocol are allowed. (Naturally, the mutex implements the “late release” protocol as its grants are mutually exclusive; the “early release” protocol can be obtained by buffering the mutex outputs.) WORKCRAFT verifies this property by checking that the following constraints are satisfied in every reachable state (the next-state value of a signal is denoted by a dash):

\[
\begin{align*}
    r_1 \cdot g_2 & \implies g_1' \\
    r_1 & \implies g_1' \\
    r_2 \cdot g_2 & \implies g_2' \\
    r_2 & \implies g_2' \\
    r_2 \cdot g_2 & \implies g_1' \\
    r_1 \cdot g_1 & \implies g_2'
\end{align*}
\]

Note that value of g1’ is implied by these constraints unless \(r_1 \cdot \overline{r_2} \cdot g_2\). The condition reflecting the flexibility of choosing between the “early release” and “late release” protocols (and symmetrically for g2’).

Interestingly, these constraints do not imply that the critical sections are mutually exclusive. That is, adding \(r_1 \cdot g_1 \cdot r_2 \cdot g_2\) to the above constraints will not lead to a contradiction, and will simply imply \(g_1' \cdot \overline{g_2'}\) and maintain the invariant \((r_1 \cdot g_1) \cdot (r_2 \cdot g_2)\) (mutual exclusion of critical sections) once it is satisfied. However, WORKCRAFT still adds an extra constraint \((r_1 \cdot g_1) \cdot (r_2 \cdot g_2)\) to check the mutual exclusion at the initial state – the rationale is that the mutex cannot be initialised in a state with both grants high, and the violation of this constraint in the initial state is very suspicious in any case.

After the STG specification has been verified, the circuit is derived by automatically factoring out arbitration into the environment, synthesising the remaining part of the controller using PETRIFY [5] or MPSat [6] backends, and automatically adding mutexes to the result.
the scenes.

This research was supported by Dialog Semiconductor and EPSRC grant EP/L025507/1 (A4A).

ACKNOWLEDGEMENTS

This research was supported by Dialog Semiconductor and EPSRC grant EP/L025507/1 (A4A).

REFERENCES