Journal of Applied Logic 15 (2016) 69–93

Contents lists available at ScienceDirect

Journal of Applied Logic

www.elsevier.com/locate/jal

A multiple-valued logic approach to the design and verification of hardware circuits

Amnon Rosenmann

Institute of Discrete Mathematics, Graz University of Technology, Steyrergasse 30, A-8010 Graz, Austria

ARTICLE INFO

Article history:
Received 21 June 2015
Accepted 29 December 2015
Available online 28 January 2016

Keywords:
Multiple-valued logic
Hardware verification
Hardware simulation
Verification complexity
De Morgan Canonical Form

ABSTRACT

We present a novel approach, which is based on multiple-valued logic (MVL), to the verification and analysis of digital hardware designs, which extends the common ternary or quaternary approaches for simulations. The simulations which are performed in the more informative MVL setting reveal details which are either invisible or harder to detect through binary or ternary simulations. In equivalence verification, detecting different behavior under MVL simulations may lead to the discovery of a genuine binary non-equivalence or to a qualitative gap between two designs. The value of a variable in a simulation may hold information about its degree of truth and its "place of birth" and "date of birth". Applications include equivalence verification, initialization, assertions generation and verification, partial control on the flow of data by prioritizing and block-oriented simulations. Much of the paper is devoted to theoretical aspects behind the MVL approach, including the reason for choosing a specific algebra for computations and the introduction of the notions of De Morgan Canonical Form and of verification complexity of Boolean expressions. Two basic simulation-based algorithms are presented, one for satisfying and verifying combinational designs and the other for equivalence verification of sequential designs.

1. Introduction

The verification and analysis of digital hardware (HW) circuits [22] has long become a major challenge during the design process. While formal verification methods such as model checking of properties [10] and formal equivalence checking [18,27] are complete, they can only be applied to designs of limited size. The traditional method of verification through simulations is incomplete, however it can be applied to larger designs. Hybrid verification methods, which combine concrete or symbolic simulations with formal methods, are also common [3]. In simulations based on ternary logic (see e.g. [12]) the binary domain is extended to include a "don't care" (or "unknown") value X. It is also common to perform simulations based on quaternary logic, which include a fourth "high-impedance" Z value. Such logics are also used for abstracting

E-mail address: rosenmann@math.tugraz.at.

http://dx.doi.org/10.1016/j.jal.2016.01.001
1570-8683/

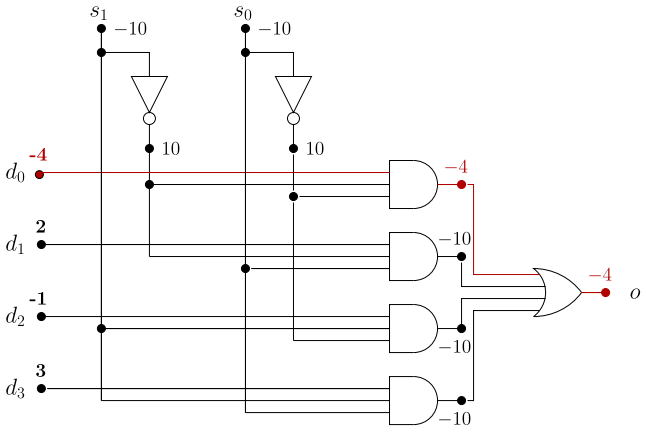

Fig. 1. Multiplexer computation.

symbolic simulations [3,38,39], in the model checking technique Symbolic Trajectory Evaluation (STE) [35], in the initialization phase and in equivalence verification [32].

In this paper we suggest to extend the ternary-based simulations methodology by using Multiple-Valued Logic (MVL). The idea is to transform the ternary domain into the domain $\widehat{\mathbb{Z}}$, an extension of the set of integers with $\pm\infty$, and transform the AND, OR and NOT gates into the minimum, maximum and negation operators respectively, that is, to adopt the semantics of the standard fuzzy operators (Zadeh operators). In order to illustrate this idea, let us look at the Multiplexer drawn in Fig. 1 in the MVL setting. The selectors s_0, s_1 are assigned negative values, corresponding to the assignment "00" in binary logic, thus the data signal d_0 is selected.

This extension to MVL is simultaneously of a refinement and of an abstraction nature. The refinement is expressed among others, in the form of "logical strength" and stability of the outcome of a computation. The output value -4 in Fig. 1 is of large absolute value (with regard to the data inputs) and it means here that its corresponding binary value (F in this case) is stable to any change in value to the data signals d_1, d_2, d_3 (while keeping the other input values unchanged). The value of the output signal o reveals not only the value of the selected input data, but also its source d_0, as shown by the path of absolute value 4 from d_0 to o in Fig. 1. In general, from any internal or output signal of a combinational circuit one can draw such a line to an input signal, such that all the values along this path are of the same absolute value. Another aspect of having a wider range of values in MVL is that it allows us to distinguish between designs that are binary equivalent. In some cases such a distinction refers to a quality gap. In other cases it can hint to the existence of a binary non-equivalence, which may be difficult to detect in the binary setting. Since non-equivalence in the MVL setting is easier to find, we can search in the near environment of an MVL non-equivalence for a "genuine", i.e. binary, non-equivalence. We present an algorithm which is based on these ideas (Algorithm 2).

Unlike simulations done in ternary logic, in the more informative MVL setting the boundary between the "care" and "don't care" domains need not be determined in advance but rather is dynamic and set upon each simulation according to the output value. This property (as stated in Theorem 3.4) allows us to raise the abstraction level and is a key factor in applying MVL to the verification of binary designs. For example, the "strong" output in the computation shown in Fig. 1 means that we can increase the abstraction level by projecting this computation to a ternary logic, while mapping the values of the input signals d_1, d_2, d_3 to "don't care" (instead of mapping every positive value to T and every negative value to F).

When performing simulations on sequential designs we can observe the change in values of a specific signal along time. In MVL simulation the dual is also possible: we can observe the change in space of a specific $\widehat{\mathbb{Z}}$-value along time. The input values may be augmented with timestamps that refer to their "date of birth". Then, when examining a state of a simulation sequence, the values of the (input, output or internal)