A step towards reconciling GALS industrial design with formal verification

Fatma Jebali

Inria
Univ. Grenoble Alpes, LIG, F-38000 Grenoble, France
CNRS, LIG, F-38000 Grenoble, France

Keywords: GALS systems, concurrent languages, formal description techniques, automatic verification

The ever-increasing applications combining both computer software and hardware together with the use of communication technologies such as Internet and mobile computing have led to new generations of heterogeneous systems called GALS (*Globally Asynchronous, Locally Synchronous*) systems. Instances of GALS systems range from multi-platform and software-based systems (among which distributed smart control systems and cyber-physical systems) to single-platform and hardware-based systems (among which heterogeneous multicore systems and networks on chip). Such systems can be extremely safety-critical.

A GALS system is composed of several synchronous subsystems, executing and interacting in asynchronous concurrency. Each subsystem is composed of several deterministic components running together in synchronous concurrency. As such, GALS systems involve a high degree of synchronous and asynchronous concurrency and are likely to be nondeterministic. Designers are forced to deal with design concepts, synchronous and asynchronous computations, deterministic and nondeterministic behaviours, and verification approaches. This makes the design of such systems increasingly complex, error-prone, and unmastered.

The integration of formal methods in the design process is crucial to help designers master that complexity and build strong confidence in the correctness of such systems. In the last few decades, formal and automatic verification based on state space exploration has proven one of the most powerful method to deal with the design of complex concurrent systems and to ensure their correctness.

Most of the existing approaches trying to verify GALS architectures adopt specific techniques of either the synchronous or the asynchronous paradigm and target a specific instance of GALS systems. In both cases, either synchrony or asynchrony is disadvantaged since not recognized as a main paradigm, narrowing down verification results accuracy and making them unsuitable to other GALS instances. Another trend has been to design new GALS-specific languages and tools to enforce the assumptions of the GALS paradigm. However, existing work either suffer from rigorous support for formal verification or require too much expertise about the GALS-paradigm and concurrency theory from users.

The current state-of-the-art is then not sufficiently advanced to handle the challenging complexity of modern GALS systems. A more generic approach is

desirable and should be: (1) able to handle a large range of GALS systems, (2) easy-to-use enough to reduce the design effort and help developing correct systems, and (3) formal to allow for automatic verification. To fulfill these requirements, we propose a GALS-dedicated framework to enhance the design process of GALS systems with automatic verification. Our primary concern is to facilitate the connection of industrial environments for designing GALS systems (which are not necessarily formal) to mostly academic formal verification tools.

The first step towards this ambitious task is the definition and development of GRL (*GALS Representation Language*), a new language with user-friendly syntax and formal semantics, to specify the behaviour of GALS systems. GRL combines synchronous features of dataflow languages and asynchronous features of process algebras (particularly LNT, an asynchronous concurrent language inheriting process algebraic concepts and extended with data and control structures). This makes possible a versatile, modular description of synchronous subsystems, environment constraints, and asynchronous communications. GRL is sufficiently expressive and general-purpose to model a wide range of GALS architectures, implemented on single or distributed platforms, and involving point-to-point or multi-point communications. Moreover, its user-friendly syntax and abstraction level, which is close to the dataflow model used in industry, makes GRL easier to learn and employ than a full-fledged process algebraic language.

GRL can independently be connected to verification frameworks based on either the synchronous or the asynchronous paradigms. The language is currently equipped with an automated translator to LNT, the input language of the CADP verification toolbox, which comprises tools for visual checking, model checking, and equivalence checking. This makes possible the analysis of GRL descriptions using the rich functionalities of the CADP toolbox (e.g., simulation, verification, performance evaluation), focusing on the asynchronous behaviour of the GALS.

GRL and the GRL2LNT translator start to be used in the Bluesky industrial project led by *Schneider Electric*. Our aim is the validation of networks of *em4*: the smart generation of *Programmable Logic Controller*, which are software-based and multi-platform control devices communicating asynchronously. The validation services will be deployed in mode SaaS (*Software As a Service*), promoting a rigorous approach to design distributed applications in the Internet of Things. After a positive feedback received from our industrial partners, we are investigating an automated connection between their *em4* design software and GRL, which would provide a complete analysis chain having CADP as verification back-end. We also develop GRL off-the-shelf programs describing function blocks, environments, and communication protocols used in em4 networks.

We plan to continue our work by applying equivalence checking and model checking techniques to industrial GALS systems described in GRL. Hardware/-software co-simulation is also possible using the EXEC/CAESAR framework of CADP, which enables the C code generated from a GRL description to be integrated with a physical platform. We also plan to investigate the connection of GRL to verification frameworks based on the synchronous paradigm to analyse the behaviour of individual blocks corresponding to synchronous subsystems.