Asynchronous design using the DISP programming language and the tools di2pn and petrify

Mark B. Josephs and Dennis P. Furey

Abstract:

The language of Delay-Insensitive Sequential Processes (DISP), a variant of Hoare’s Communicating Sequential Processes and of Josephs and Udding’s DI-Algebra, may be attractive to designers of asynchronous logic blocks, particularly for controllers that use request and acknowledge signals. Programming in DISP is convenient for exploring a variety of handshaking protocols, an important aspect of asynchronous design. Small changes to the order in which signal transitions (from logic-0 to logic-1, or vice versa) occur in a program (called “reshuffling” by Martin) can have a significant impact on area and performance when the program is implemented as an asynchronous circuit. Such changes may inadvertently introduce deadlock or affect the complexity of logic synthesis.

DISP is a structured, parallel programming language, but without program variables and assignment. Instead, input/output-bursts (as in Nowick and Dill’s “Burst-Mode specifications”) serve as primitive statements, where inputs and outputs are to be interpreted as signal transitions. DISP is similar to Martin’s “handshaking expansions” (HSE), but more uniform in its treatment of signals.

A DISP specification consists of a pair of programs, one describing the behaviour of the logic block and the other describing the behaviour of the environment in which it will operate. Publicly available at http://www.sbu.ac.uk/~fureyd/di2pn/ and http://www.lsi.upc.es/~jordic/petrify/, the CAD tools di2pn and petrify can be used to automatically validate a DISP specification and to automatically synthesise asynchronous logic from it. di2pn is used at the front end, translating the specification into a Petri net. It uses the same text-file format as petrify, an enhanced version of the ASTG format devised for SIS. It is petrify that does the actual validation and logic synthesis, interpreting the Petri net as a Signal Transition Graph.

In this talk, we apply the DISP programming language, and the tools di2pn and petrify, to the design of a number of interesting asynchronous logic blocks that can be found in the literature.