High-level synthesis of asynchronous circuits from control data flow graph representations.

S. F. Nielsen¹ \quad J. Sparso¹ \quad J. Madsen¹ \quad J. P. Hammerstoft¹ \quad J. S. Jensen²

¹) Informatics and Mathematical Modeling
   Technical University of Denmark
   Richard Petersens Plads, Bldg. 322
   DK-2800 Kgs. Lyngby, Denmark
   e-mail: {sf, jsp, jan}@imm.dtu.dk

²) MIPS Denmark Development Center
   Lautrupvang 4B
   DK-2750 Ballerup, Denmark

We present the initial work on a high-level behavioral synthesis tool for asynchronous circuits. Our current tool can synthesize a single VHDL process (assuming inputs and outputs to be handshake channels) into a standard cell circuit implementation. The current tool is capable of performing non-performance degrading resource sharing as well as performance degrading resource sharing.

Our goal is to apply techniques from high-level synthesis of synchronous circuits to the synthesis of asynchronous circuits. Our synthesis flow follows these steps:

- An internal representation – in the form of a Control Data Flow Graph (CDFG) – is extracted from the specification. A CDFG captures only the control and data dependencies that are inherent in the computation. In this way it is not biased towards a certain implementation.

- The CDFG is analyzed and resource sharing and operation scheduling, in the form of graph transformations, is performed.

- A corresponding circuit implementation is generated. Currently we use the fact that there is a close correspondence between a CDFG [18, 15, 7] and an asynchronous circuit: The edges in a CDFG can be seen as handshake channels and the nodes in a CDFG can be seen as handshake components – components that are quite similar to the handshake components used in syntax directed compilation. In this way we perform a simple one-to-one mapping of the CDFG to a network of asynchronous handshake components.

The graph transformations makes this different from the syntax directed compilation of large-scale asynchronous circuits from non-standard languages (typically CSP-like) where the compiler merely transforms a program into a corresponding circuit: Tangram [16, 3, 13], OCCAM [5, 4], Balsa [1, 2, 8].

References


