“A Burst-Mode Oriented Back-End for the Balsa Synthesis System”

Steven M. Nowick  
Columbia University  
New York, NY, USA  
(joint work with Tiberiu Chelcea, Andrew Bardsley, and Doug Edwards)

ABSTRACT:

This presentation describes a new integrated back-end to the University of Manchester’s Balsa Synthesis System.

The Balsa System is now becoming widely used, and allows for the high-level modelling and synthesis of large-scale asynchronous systems. The approach has been effectively applied to processors and substantial custom components. However, a key limitation of the synthesis method is that it is “syntax-directed”: each language construct is translated directly to a corresponding intermediate ‘handshake component’, which is in turn mapped to a circuit implementation. As a result, there is little ‘design-space exploration’ or aggressive use of optimizations, and the result can be significant area and performance penalties.

In this presentation, several new contributions are proposed, to overcome these bottlenecks:

• optimization algorithms:

  Two new clustering algorithms are proposed. Each takes as input the (unoptimized) Balsa netlist of handshake components, and merges components into larger but moderate-sized clusters. Each cluster is then synthesized using a burst-mode CAD tool, MINIMALIST.

• new controller specification language:

  A small channel-based description language, called CH, is introduced, to model handshake components. This language is intermediate between higher-level description languages (e.g. Balsa, Tangram) and low-level languages which specify individual signal transitions. The language is effective for manipulating and composing handshake components.

• CAD package:

  The new optimizations are implemented in a CAD tool, which is integrated into the existing Balsa flow, along with Balsa-to-CH and CH-to-Burst-Mode translators. Synopsys’ Design Compiler is used for technology mapping.

Initial experiments on four substantial examples were performed:

the SSEM asynchronous 32-bit processor core, a systolic counter, a wagging register, and a stack. Performance improvements up to 21% were obtained.

While the above initial clustering optimizations are limited to control components, we will also outline some of our very recent work on more powerful peephole and clustering optimizations, including on datapath components, which achieve even better performance improvements.