Compiling the language Balsa to delay insensitive hardware

Andrew Bardsley, Doug Edwards
Department of Computer Science
University of Manchester, Manchester, M13 9PL, UK,
phone: +44 161 275 6844, fax: +44 161 275 6204,
e-mail: **bardsley@cs.man.ac.uk**, **doug@cs.man.ac.uk**

Abstract

A silicon compiler, Balsa-c, has been developed for the automatic synthesis of asynchronous, delay-insensitive circuits from the language Balsa. Balsa is derived from CSP with similar language constructs and a single-bit granularity type system.

Balsa compiles to intermediate handshake circuits by an extended form of the compilation function used in the Tangram system. The handshake circuits are subsequently mapped to CMOS implementations of 4-phase bundled-data asynchronous circuits by a suite of parameterised component-generating scripts within the Cadence design framework.

Keywords

High-Level Synthesis, Design Systems and Tools

1 INTRODUCTION

In the past 5 years there has been a resurgence of interest in asynchronous design methodologies prompted by the increasing problems encountered by designers of synchronous systems in the areas of clock skew, EMC, and power dissipation. Asynchronous systems offer solutions to these difficulties whilst also providing modularity (an increasingly attractive feature with the emergence of 'coreware' products) and the opportunity to exploit average case performance.

The credibility of the asynchronous approach has been enhanced by work at Philips (van Berkel, Rem 1993) where the Tangram silicon compiler has produced a DCC decoder with substantial power savings and by the work of the AMULET group at Manchester University.

In spite of the now proven feasibility of asynchronous techniques, their widespread acceptance by the wider design community is likely impeded by the lack of automated synthesis or peculiarly asynchronous support tools. A number of STG based tools such as ASSASSIN, PETRIFY, FORCAGE do exist but their utility lies mainly in the synthesis of relatively small blocks of control logic. Apart from the the proprietary

Tangram system of Philips, there is a dearth of high-level tools available. The absence of such tools has been keenly felt during the development of the AMULET processors and has prompted a number separate tool based projects at Manchester (Rainbow (Barringer *et al* 1996), Lard and balsa).

This paper describes Balsa, a Tangram-like language which compiles to an intermediate set of handshake components. This development arises from previous work undertaken as part of the EU funded EXACT project (ESPRIT 6143) in which the Tangram tool was embedded within the Cadence design system to allow interaction by engineers at both the handshake level and the flattened netlist level.

2 THE BALSA LANGUAGE

Balsa is an imperative language implementing a CSP (Hoare 1985) like synchronisation scheme. CSP regards a communication between concurrently executing processes to be synchronising action where all participants must be willing to enter into the communication before it can take place, in CSP this involves two named processes (one performing an input and the other an output). Balsa extends this principle to allow one to many communications (one write to many reads) and communications between ports on processes rather than the CSP notion of type constructors. The following is an example of a 16 entry register file with a single read and single write ports:

```
type word is 16 bits
type reg_sel is 4 bits

procedure register_bank (
    input w : word; output r : word;
    input rNw : boolean; input sel : reg_sel ) is
local
    variable R : array over reg_sel of word
    variable rNw_var : boolean
    variable sel_var : reg_sel
begin loop
        rNw -> rNw_var || sel -> sel_var;
        if rNw = true then r <- R[sel_var]
                else w -> R[sel_var] end
end end
```

The Balsa language possesses the following features:

- One-to-many communicating channels, with explicit arbitration
- Statically checked shared data
- Fine grain parallelism

3 THE BALSA-C COMPILER

Balsa-c is a single pass compiler capable of transforming a valid Balsa program into a network of handshake components. The resulting handshake component netlist is transformed into delay insensitive hardware using the parameterised component generating backend produced by the EXACT project.

Handshake components communicate solely by means of interconnecting channels on which data validity/synchronisation is managed by the imposition of a handshaking protocol. In the current compilation system a 4-phase broad protocol is employed although other protocols (or indeed a chosen mixture of protocols) may be used in future in order to optimise for speed/area.

4 CONCLUSION & FURTHER WORK

Balsa and the Balsa-c compiler serve to produce a direct mapping from program to handshake circuits with the minimum of transformation.

Future work involving Balsa will be concerned with the need to optimise the output handshake component networks both independent of and in respect of a specific target technology. It is aimed to produce optimising methodologies which result in not only smaller, faster circuits but which also preserve the attractively direct mapping from language constructs to target hardware.

Increasing the expressive range of Balsa to improve the size/speed of unoptimised output is also an aim.

5 REFERENCES

Barringer, H., Fellows, D., Gough, G., Jinks, P., Marsden, B., Williams, A. (1996), A Framework for Asynchronous Micropipeline Circuits. *in* Proceedings of European Simulation Symposium (ESS'96) – Genoa Italy

Hoare, C. (1995) *Communicating Sequential Processes*, Prentice-Hall

van Berkel, K., Rem M. (1993), VLSI Programming of Asynchronous Circuits for Low Power. *in* 'Asynchronous Digital Circuit Design' Proceedings of Asynchronous Design Workshop – Banff Alberta, Birtwistle, G., Davis, A. editors, Springer-Verlag

6 BIOGRAPHY