An Imperative Asynchronous Language for Kahn Semantics in the spirit of Balsa/Tangram

COMS 4115 Whitepaper - February 2005

Clive John Chuan
clive@chez.com

Jane Sui-Tit-Tong
jane_sui@yahoo.co.uk

ABSTRACT

This whitepaper gives an overview of the definition of a high-level asynchronous language and the implementation of the compiler for defining the handshake paradigm for concurrent processes.

1. INTRODUCTION

The hardware design world is dominated by the use of a global clock to synchronize components in an integrated circuit. Demand for low power portable devices has renewed interest in clockless chip design, since the increasingly fast clocks tend to dissipate more heat and the distribution of the clock to all parts of the circuit becomes more difficult. Asynchronous design instead relies on handshaking between components as the means of communication between components. This handshaking paradigm can be likewise be used for concurrent processes. The various applications of this paradigm could span from parallel computing to embedded systems, where there is a strong relationship between software and hardware.

2. RELATED WORK

The current project is based on the use of asynchronous hardware systems. Hardware systems have been predominently been relying on a single system clock which serves to define when the different components communicate with each other. With processor speed increasing ever more, it is becoming increasingly difficult to synchronize all the components and ensure the accuracy of the propagation of the clock across the chip layout. Some hardware researchers have therefore been working on systems which do not rely on a clock, but instead communicate asynchronously. The Tangram research project at Philips has created a language and tools to compile Tangram procedures into VLSI circuits using the handshake components as the communication links between components. The Balsa project from the University of Manchester has also created a Tangram-like language definition, Balsa. The current project will closely follow the Balsa language, in particular the handshaking between components and map these to concurrent software processes.

3. GOALS

The main goal of this project is to define a simple language for imperative asynchronous language, derived from the Balsa hardware description language. We also create a compiler for this language, which would take as input the procedures defining concurrent processes and the communication patterns between them. The output of the compiler would be Java code that would implement the underlying concurrency classes and provide the handshaking in Java in a transparent way to the user.

4. HANDSHAKE COMPONENTS

Handshake components are components that are connected by handshake communicating channels. The handshake circuits were introduced by van Berkel [6] as a representation for asynchronous circuits in the Tangram hardware description language. Each handshake channel connects a passive and an active port of 2 handshake components.

We define a handshake to be a pair of request from one component and an acknowledgement from the other component. An active port is the port that initiates a request and a passive port is on the receiving end of the channel and therefore sends back the acknowledgement. A pair of request and acknowledgements on a handshake channel is called a sync, by which 2 components can synchronize each other. A handshake channel can also be used to transfer data from one component to the other. This can be defined as Push and Pull communications, where the direction of data flow and the direction of the request differentiates between the 2 types of communication. Push communications exist when the data follows the same direction as the request, i.e., either when an active port is sending data or when the passive port is receiving data. On the other hand, Pull communications are defined when the data is in the same direction as the acknowledgement: an active port receives the data or a passive port sends the data in the acknowledgement.

5. SAMPLE CODE

Below is a sample code for a single-place buffer as described in the Balsa Tutorial [1]. This is the description of a byte-wide single place buffer.

```
procedure buffer1 (input i : byte; output o : byte) is
  variable x : byte
begin
  loop
    i -> x  // putting input into variable x
    ;       // sequential
    o <- x  // sending output from variable x to output
  end
end
```

The code above puts the input channel `i` into the variable `x` when the latter is ready to accept the data. The ; here means that the operations have to be done sequentially. Afterwards, the variable `x` is sent to the environment via the output channel `o`.
The corresponding Java code that is compiled from this in-

put would therefore implement the handshake channels into classes and methods.

6. REFERENCES

[1] D. Edwards, A. Bardsley, L. Janin, W. Toms. Balsa: A Tutorial Guide version V3.4.2. Jan 2005.

[2] A. Bardsley. Balsa: An Asynchronous Circuit Synthesis System. Masters Thesis, Department of Computer Science, The University of Manchester, 1998.

[3] A. Bardsley. *Balsa: An Asynchronous Circuit Synthesis System.* PhD. Thesis, Department of Computer Science, The University of Manchester, 1998.

[4] A. Peeters and K. van Berkel. Synchronous handshake circuits. In *Proceedings of the Seventh International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 86–95, Salt Lake City, Utah, Mar. 2001.

[5] K. van Berkel. *Handshake Circuits: An Intermediary Between Communicating Processes and VLSI.* PhD thesis, Eindhoven University of Technology, The Netherlands, May 1992.

[6] K. van Berkel. *Handshake Circuits: An Asynchronous Architecture for VLSI Programming.* Cambridge University Press, 1993.

[7] K. van Berkel, J. Kessels, M. Roncken, R. Raeijs, and F. Schalij. The VLSI-programming language Tangram and its translation into handshake circuits. In *Proceedings of European Design Automation (EDAC)*, pages 384–389, Amsterdam, The Netherlands, Feb. 1991.

[8] M. B. Josephs, S. M. Nowick, and C. H. K. van Berkel. Modeling and design of asynchronous circuits. *Proceedings of the IEEE*, 87(2):232–242, Feb. 1999.