Functional Hardware Design in CλaSH

Christiaan Baaij, Rinse Wester, Anja Niedermeier, Arjan Boeijink, Marco Gerards, Jan Kuper, Gerard Smit - Compute Architectures for Embedded Systems group, University of Twente, Enschede, The Netherlands

CλaSH [1, 3] is a compiler system for generating hardware as described by a mathematical/functional specification of the architecture. The specification language is a subset of the functional programming language Haskell and offers:

Polymorphism: a hardware component that is useful for different types of data has to be specified only once.

Higher-order functions: regular structures can be expressed very concisely and are well readable.

Lambda abstraction: purely combinational circuits can be expressed in-line.

Pattern matching: definition-by-cases has a compact format and is well readable.

Functional composition: composition of components exploits signals directly without being bothered by port mappings.

Type derivation: types of input and output signals of nested components can be derived automatically from the types of the outer component.

Functional languages are especially well suited to describe hardware because combinational circuits can be directly modeled as mathematical functions and functional languages are very capable of describing and (de-)composing these functions.

The CλaSH system uses a rewrite mechanism that exhaustively applies meaning-preserving transformations on the specification to generate RTL-style VHDL. Several cases have shown that the generated designs have similar size and operating frequency as designs hand-written in RTL-style VHDL. We will demonstrate these designs, a reduction circuit for Sparse Matrix Vector multiplication (SMxV) [1, 2] and a Data-flow processor [4], including their corresponding quality parameters such as size, energy usage, and operating frequency.

To show that CλaSH is well-suited to specify a mixture of data- and control-oriented hardware we will also demonstrate a music synthesizer and a spectrum analyzer. The designs are synthesized for an Altera Cyclone II FPGA development board, and interface with several of the peripherals on this board such as an audio CODEC and a VGA controller. The design of this system also demonstrates the use of multiple clock domains in a CλaSH architecture specification.

`http://clash.ewi.utwente.nl/`

References

[1] C.P.R. Baaij, M. Kooijman, J. Kuper, W.A. Boeijink, M.E.T. Gerards, CλaSH: Structural Descriptions of Synchronous Hardware using Haskell. In: *Proceedings of the 13th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools (DSD 2010)*, Lille, France, 1-3 Sep 2010, pp. 714–721.

[2] M.E.T. Gerards, J. Kuper, A.B.J. Kokkeler, E. Molenkamp, Streaming Reduction Circuit, In: *Proceedings of the 12th EUROMICRO Conference on Digital System Design, Architectures, Methods and Tools (DSD 2009)*, Patras, Greece, 2009, pp. 287–292

[3] J. Kuper, C.P.R. Baaij, M. Kooijman, M.E.T. Gerards, Exercises in architecture specification using CλaSH, In: *Proceedings of Forum on Specification and Design Languages (FDL 2010)*, Southampton, UK, pp. 178–183.

[4] A. Niedermeier, R. Wester, C.P.R. Baaij, J. Kuper, G.J.M. Smit, Comparing CλaSH and VHDL by implementing a dataflow processor. In: *Proceedings of the Workshop on PROGram for Research on Embedded Systems and Software (PROGRESS 2010)*, Veldhoven, The Netherlands, pp. 216–221.

Example: FIR Filter

```
module FIR where

import CLasH.HardwareTypes

-- Polymorphic Dot-product defined using higher-order functions
-- Works for all vector lenghts and all number types
dotp as bs = vfoldl (+) 0 (vzipWith (*) as bs)

-- N-tap polymorphic FIR Filter:
-- hs : Filter coefficients
-- pxs: previous inputs (registers)
-- x  : current input
fir hs (State pxs) x = (State (x +>> pxs), dotp pxs hs)

-- 4-tap, 16-bit signed integer, FIR Filter:
-- coefficients: <2,3,-2,8>
-- registers initialized to 0
fir4 = (fir coeffs) ^^^ initR
  where
    coeffs = $(vTH [2,3,-2,(8::Signed D16)])
    initR  = vcopyn d4 0
```