Chext: A Domain-specific Language for Safe and Agile Elastic Dataflow Accelerators

Canberk Sönmez
EPFL
Lausanne, Switzerland
canberk.sonmez@epfl.ch

Mohamed Shahawy
EPFL
Lausanne, Switzerland
mohamed.shahawy@epfl.ch

Paolo Ienne
EPFL
Lausanne, Switzerland
paolo.ienne@epfl.ch

Abstract

Creating dataflow circuits with HLS tools to exploit memory- or task-level parallelism is challenging. These tools can generate functional designs easily, but optimizing them for performance often requires extensive code restructuring, pragma tuning, and even RTL integration. Writing low-level RTL, however, is rarely productive. We propose Chext, a domain-specific language that provides a thin abstraction layer over RTL for building complex dataflow circuits with elastic interfaces. Chext enables a safe and agile development flow with its novel approach and offers a rich component library to accelerate design, making such circuits more accessible on FPGAs. In this paper, we introduce Chext, outline its design principles, and present two case studies comparing it with Vitis HLS.

CCS Concepts

• **Hardware → High-level and register-transfer level synthesis**; *Hardware accelerators*; • **Software and its engineering → Domain specific languages**.

Keywords

Field-Programmable Gate Arrays, High-Level Synthesis, Domain-Specific Languages

ACM Reference Format:
Canberk Sönmez, Mohamed Shahawy, and Paolo Ienne. 2026. Chext: A Domain-specific Language for Safe and Agile Elastic Dataflow Accelerators. In *Proceedings of the 2026 ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '26), February 22–24, 2026, Seaside, CA, USA.* ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/3748173.3779543

1 Motivation

Achieving high performance on FPGAs frequently demands precise control over dataflow, backpressure, and memory behavior. While top-down design flows can be productive for functional prototypes, they often require extensive refactoring, heuristic tuning, or mixed flows to reach performance targets—especially for memory-bound workloads and task-/memory-level parallelism. Conversely, purely bottom-up RTL offers full control but can be slow, error-prone, and cumbersome for building and verifying complex elastic pipelines.

Elastic interfaces provide a compelling middle ground for a bottom-up construction of accelerators: components can be composed schematically, handshakes can be reasoned about mechanically, and timing decoupling emerges naturally. This paradigm has, arguably, enabled high-throughput designs in prior work; however, a practical, RTL-faithful, and ergonomic toolchain for designers who want to construct such systems directly has been lacking.

Chext aims to fill this gap. It provides a Chisel-based elastic-first DSL that lets designers assemble dataflow circuits from simple, predictable building blocks; reason explicitly about flow control, resource sharing, and state; and, importantly, build memory subsystems that are tailored to workload needs.

2 What is Chext?

Chext is a Scala/Chisel-based domain-specific language for constructing safe and agile elastic dataflow accelerators on FPGAs. It offers a thin abstraction over RTL so that implementations closely mirror the intended schematic, while still benefiting from parametric generators, static checks, and simulation-time assertions.

Key properties

- *Elastic-first*: ready/valid handshakes, eager forks and lazy joins, queues/buffers.
- *Composable and RTL-faithful*: circuits are written as a textual representation of their dataflow graphs; components do exactly what they state, enabling predictable performance tuning.
- *Memory-aware*: explicit and tunable memory behavior using AXI4 components (burst/non-burst read/write), width converters, response/write buffers, and customizable interconnects (mux/demux).
- *Stateful when needed*: transducers (state machines), counters, fold/reduce, loop (for/while-like), scope (single-token regions).
- *User signaling*: configurable data-is-control fields, such as *first/last*, *index*, *valid/skip/zero*, *destination/select/tag*, *operand*, and *user*.
- *Safety and debugging*: static checks and simulation-time assertions can reveal handshake or protocol mistakes early.
- *Agile*: faster iteration on complex dataflow topologies (fork/join, mux/demux/arbiter, resource sharing).

Availability

- *Chext repo*: https://github.com/hdlstuff/chext
- *Getting started*: https://github.com/hdlstuff/hdlstuff
- *Code examples*: https://github.com/hdlstuff/chext-examples
- *Contact*: `canberk.sonmez@epfl.ch`

ACM ISBN 979-8-4007-2079-6/2026/02
https://doi.org/10.1145/3748173.3779543