Experience Report:

Translating Haskell / Clash to Coq / Agda / Liquid Haskell

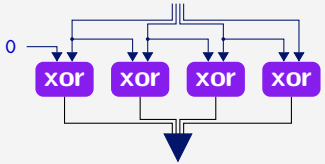
Felix Klein

HIW – June 6th, 2025

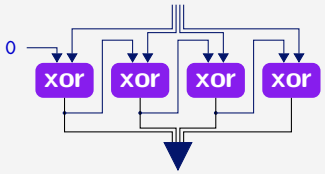
QBayLogic.
The key to FPGA-ASIC design

Running Example: Gray Code

4-bit Binary-to-Gray conversion



4-bit Gray-to-Binary conversion

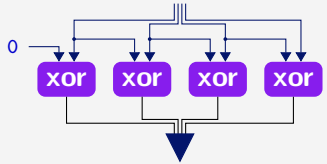


Binary-to-Gray ∘ Gray-to-Binary ≡ Identity

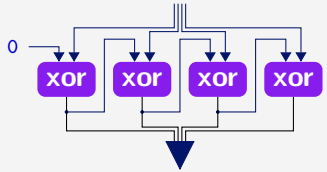
$\mathbb{N}$	Binary Encoding	Gray Encoding
0	0 0 0 0	0 0 0 0
1	0 0 0 **1**	0 0 0 **1**
2	0 0 **1** 0	0 0 **1** **1**
3	0 0 **1** **1**	0 0 **1** 0
4	0 **1** 0 0	0 **1** **1** 0
5	0 **1** 0 **1**	0 **1** **1** **1**
6	0 **1** **1** 0	0 **1** 0 **1**
7	0 **1** **1** **1**	0 **1** 0 0
8	**1** 0 0 0	**1** **1** 0 0
9	**1** 0 0 **1**	**1** **1** 0 **1**
10	**1** 0 **1** 0	**1** **1** **1** **1**
11	**1** 0 **1** **1**	**1** **1** **1** 0
12	**1** **1** 0 0	**1** 0 **1** 0
13	**1** **1** 0 **1**	**1** 0 **1** **1**
14	**1** **1** **1** 0	**1** 0 0 **1**
15	**1** **1** **1** **1**	**1** 0 0 0

Running Example: Gray Code

4-bit
Binary-to-Gray
conversion

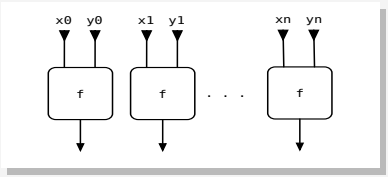


4-bit
Gray-to-Binary
conversion

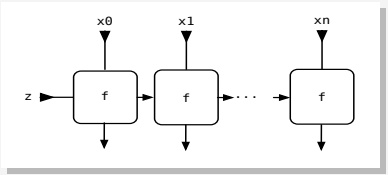


Binary-to-Gray ∘ Gray-to-Binary ≡ Identity

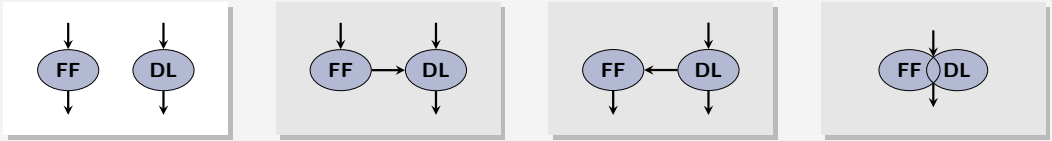
```
zipWith ::
  (a → b → c) →
  Vec n a → Vec n b → Vec n c
```




```
postscanl ::
  (b → a → b) →
  b → Vec n a → Vec n b
```



Haskell / Clash ↔ Coq / Agda / Liquid Haskell



FF: Formal Framework (e.g. proof assistant) DL: Design Language (e.g. Haskell / Clash)

Classic: Parallel Development of Implementation and Proof

✓ works without a dedicated FF ↔ DL interface
✓ expert separation of concerns

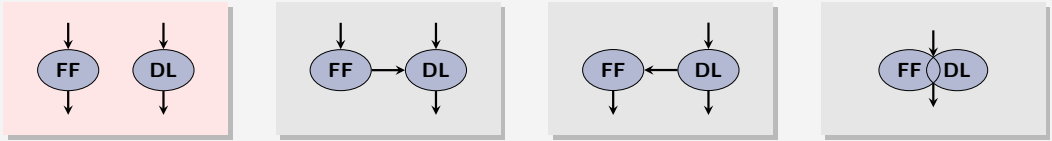
✗ error prone due to misunderstandings and miscommunication
✗ requires extensive documentation and communication
✗ counteractions the benefits from formal verification

Analyzed Approaches

Paper: *An approach to translating Haskell programs to Agda and reasoning about them*

✓ Haskell / Clash & Agda have close syntax & concepts
✗ fragile, inconvenient user experience

Haskell / Clash ↔ Coq / Agda / Liquid Haskell



FF: Formal Framework (e.g. proof assistant) DL: Design Language (e.g. Haskell / Clash)

Classic: Parallel Development of Implementation and Proof

✓ works without a dedicated FF ↔ DL interface
✓ expert separation of concerns

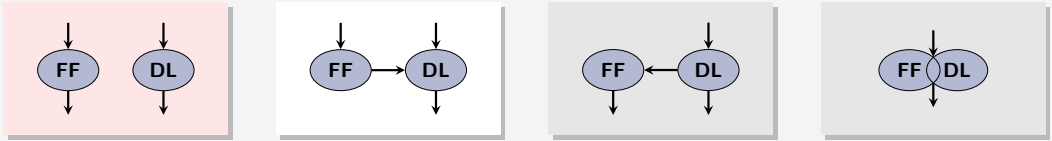
✗ error prone due to misunderstandings and miscommunication
✗ requires extensive documentation and communication
✗ counteractions the benefits from formal verification

Analyzed Approaches

Paper: *An approach to translating Haskell programs to Agda and reasoning about them*

✓ Haskell / Clash & Agda have close syntax & concepts
✗ fragile, inconvenient user experience

Haskell / Clash ↔ Coq / Agda / Liquid Haskell



FF: Formal Framework (e.g. proof assistant) DL: Design Language (e.g. Haskell / Clash)

DSL: Embed Verification Requirements of DL in FF

✓ well-defined FF → DL interface
✓ allows to make full use the FF

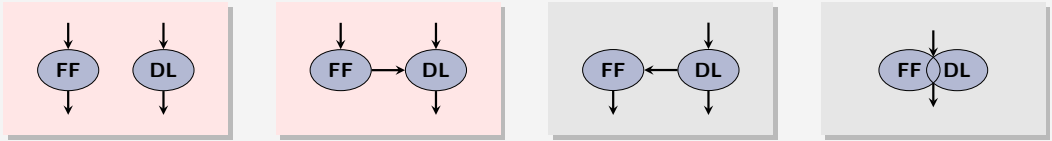
✗ every design requires some experts for FF & DL
✗ requires dedicated tooling support (design cost + maintenance)
✗ FF lock-in

Analyzed Approaches

Tool: *Alonzo* (GHC backend of Agda)
✗ designed for generating programs
✗ generates mostly unreadable code

Tool: *agda2hs* (transpiler)
✓ works with the latest GHC
✗ missing indexed datatype support

Haskell / Clash ↔ Coq / Agda / Liquid Haskell



FF: Formal Framework (e.g. proof assistant) DL: Design Language (e.g. Haskell / Clash)

DSL: Embed Verification Requirements of DL in FF

✓ well-defined FF → DL interface
✓ allows to make full use the FF

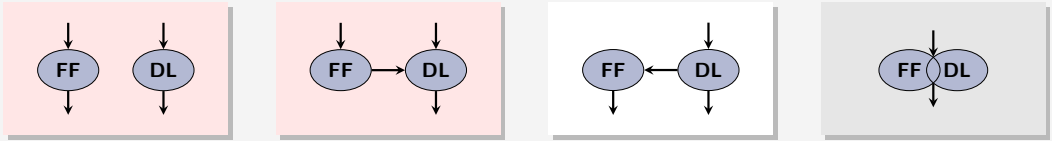
✗ every design requires some experts for FF & DL
✗ requires dedicated tooling support (design cost + maintenance)
✗ FF lock-in

Analyzed Approaches

Tool: *Alonzo* (GHC backend of Agda)
✗ designed for generating programs
✗ generates mostly unreadable code

Tool: *agda2hs* (transpiler)
✓ works with the latest GHC
✗ missing indexed datatype support

Haskell / Clash ↔ Coq / Agda / Liquid Haskell



FF: Formal Framework (e.g. proof assistant) DL: Design Language (e.g. Haskell / Clash)

Transpiler: Translate Verification Requirements between DL & FF

✓ well-defined FF ↔ DL interface
✓ expert separation of concerns

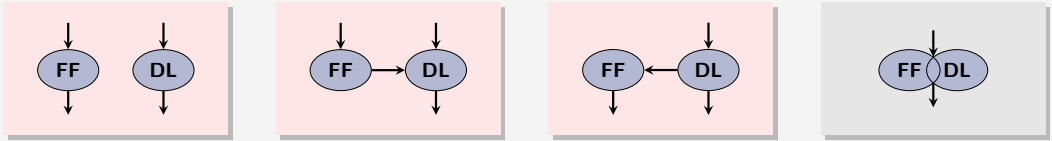
✗ requires dedicated tooling support (design costs + maintenance)
✗ restricts the design process (lock-in)
✗ missing synergies between FF and DL are critical

Analyzed Approaches

Tool: *hs-to-coq*
✓ working POC with old Clash version
✗ not maintained any more since 2021
✗ high development and maintenance cost with relative marginal gain

Tool: *haskabelle* ✗ too old (2013)

Haskell / Clash ↔ Coq / Agda / Liquid Haskell



FF: Formal Framework (e.g. proof assistant) DL: Design Language (e.g. Haskell / Clash)

Transpiler: Translate Verification Requirements between DL & FF

✓ well-defined FF ↔ DL interface
✓ expert separation of concerns

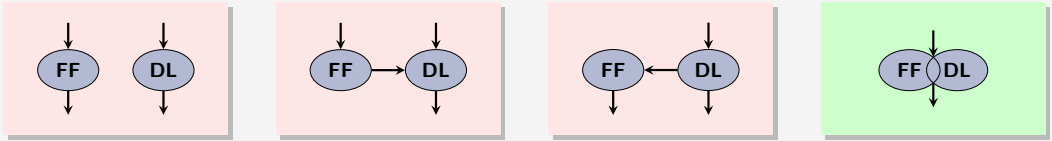
✗ requires dedicated tooling support (design costs + maintenance)
✗ restricts the design process (lock-in)
✗ missing synergies between FF and DL are critical

Analyzed Approaches

Tool: *hs-to-coq*
✓ working POC with old Clash version
✗ not maintained any more since 2021
✗ high development and maintenance cost with relative marginal gain

Tool: *haskabelle* ✗ too old (2013)

Haskell / Clash ↔ Coq / Agda / Liquid Haskell



FF: Formal Framework (e.g. proof assistant) DL: Design Language (e.g. Haskell / Clash)

Language Feature: Tightly integrate FF as part of DL

✓ well-defined FF ↔ DL interface
✓ expert separation of concerns (only experts will use the feature)
✓ single source of truth

✗ requires language (tooling) support (design cost + maintenance)
✗ restricts the design process (lock-in)

Analyzed Approaches

Tool: *Liquid Haskell*
✓ multiple working POCs
✓ works with the latest GHC
✓ SMT + Proof Combinators support
✓ Paper: *Explicit Refinement Types*
✗ only partial Clash primitive support