Comparing functional Embedded Domain-Specific Languages for hardware description

João Paulo Pizani Flor

Department of Information and Computing Sciences, Utrecht University

February 13th, 2014

Universiteit Utrecht

Table of Contents

Section 1

Introduction

Universiteit Utrecht

Hardware design

Universiteit Utrecht

Domain-Specific Languages

A computer language (turing-complete or *not*) targeting a *specific application domain.*

Example DSLs:

- SQL (database queries)
- CSS (document formatting)
- MATLAB (Matrix programming)
- VHDL (Hardware description)

Domain-Specific Languages

A computer language (turing-complete or *not*) targeting a *specific application domain.*

Example DSLs:

- SQL (database queries)
- CSS (document formatting)
- MATLAB (Matrix programming)
- VHDL (Hardware description)

A DSL can also be *embedded* in a general-purpose language.

Example EDSLs:

- Boost.Proto (C++ / parser combinators)
- Diagrams (Haskell / programmatic drawing)
- Parsec (Haskell / parser combinators)

Example of an EDSL: Parsec

A simple parser for a "Game of Life"-like input format:

```
dead, alive :: Parser Bool
dead  = fmap (const False) (char '.')
alive = fmap (const True) (char '*')

line :: Parser [Bool]
line  = many1 (dead <|> alive)

board :: Parser [[Bool]]
board = line `endBy1` newline

parseBoardFromFile :: FilePath -> IO [[Bool]]
parseBoardFromFile filename = do
    result <- parseFromFile board filename
    return $ either (error . show) id result
```


Universiteit Utrecht

Hardware EDSLs

An EDSL used for hardware design-related tasks. Can encompass:

- Modelling / description
- Simulation (validation)
- Formal verification
- Synthesis to other (lower-level) languages

Example of a hardware EDSL

Some Lava code. . .

Section 2

Analyzed EDSLs

Choice criteria

Chosen EDSLs

The language we chose to evaluate, with the respective host language, were:

- Lava (Haskell - `chalmers-lava` *dialect*)
- ForSyDe (Haskell)
- Coquet (Coq)

Evaluation criteria

- Simulation
- Verification
- Genericity
- Depth of embedding
- Tool integration
- Extensibility

Section 3

Modeled Circuits

Choice criteria

- Not too simple, not too complex
- Familiar to any hardware designer
  - No signal processing, etc.
- Well-defined, pre-specification
  - Results to verify the models against

Chosen circuits

We cherry-picked circuits from the book "Elements of Computing Systems", as they satisfied all of our demands.

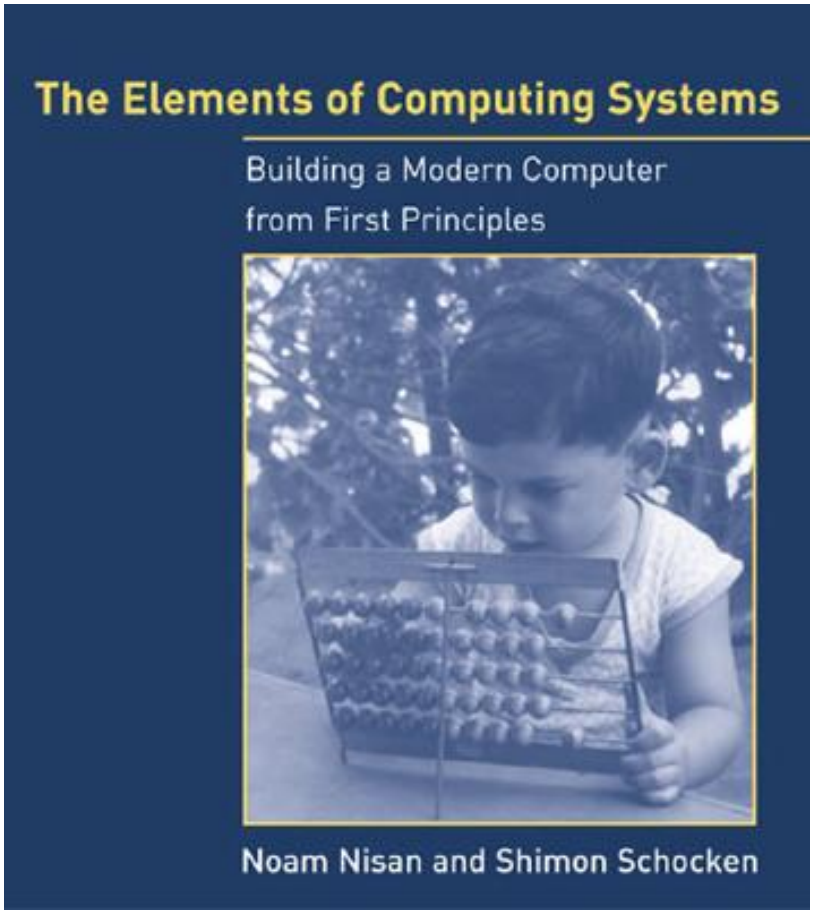

Figure: "Elements of Computing Systems" - Nisan, Schocken, available at `http://www.nand2tetris.org`.

Universiteit Utrecht

Chosen circuits

Circuit 1 A 2-input, 16-bit-wide, simple ALU

Circuit 2 A 64-word long, 16-bit wide memory block

Circuit 3 An *extremely* reduced instruction set CPU, the *Hack* CPU.

Let's take a quick look at each of these circuit's specification. . .

Circuit 1: ALU

Some of the circuit's key characteristics:

- 2 *operand inputs* and 1 *operand output*, each 16-bit wide
- 1 *output flag*
- Can execute 18 different *functions*, among which:
  - Addition, subtraction
  - Bitwise AND / OR
  - Constant outputs
  - Addition of constants to an operand
  - Sign inversion

Circuit 1: block diagram

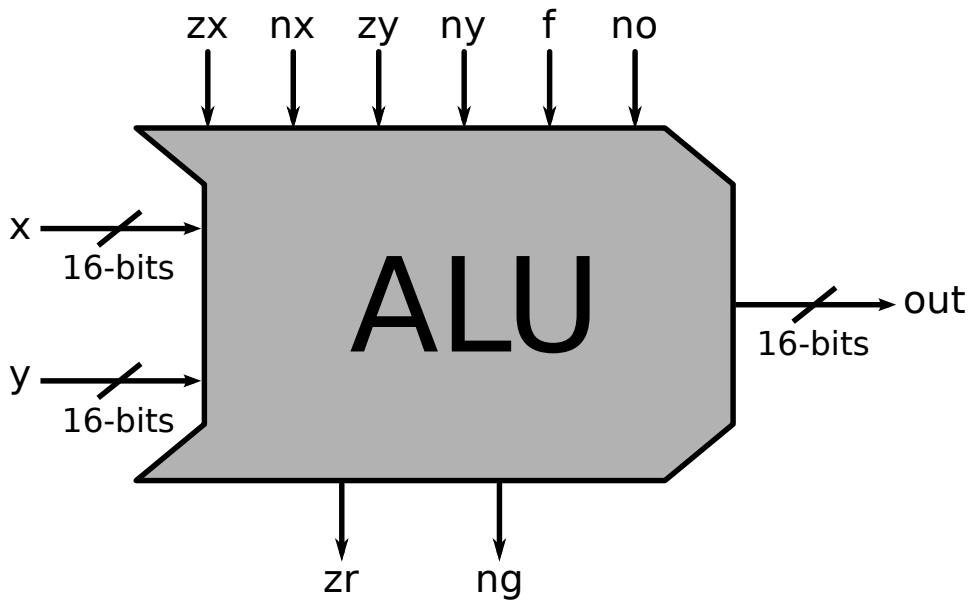

Figure: Input/Output ports of *circuit 1*, the ALU.

Universiteit Utrecht

Circuit 1: Specification

The behaviour of the ALU is specified by the values of the *control bits* and flags:

- zx and zy — *Zeroes* the x and y inputs, respectively
- nx and ny — *bitwise negation* in the x and y inputs
- f — Selects the function to be applied: $f = 1$ for addition, $f = 0$ for bitwise AND
- no — *bitwise negation* on the output ALU output
- zr and ng — The output *flag* zr $= 1$ *iff* the ALU output is zero. ng $= 1$ *iff* the output is negative.

Operations such as bitwise OR, subtraction, etc. can be done by setting the control bits appropriately.

Circuit 2: RAM64

Some of the circuit's key characteristics:

- *Sequential* circuit, with clock input
- 64 memory words stored, each 16-bit wide
- Address port has width $\log_2 64 = 16$ bit

Circuit 2: block diagram

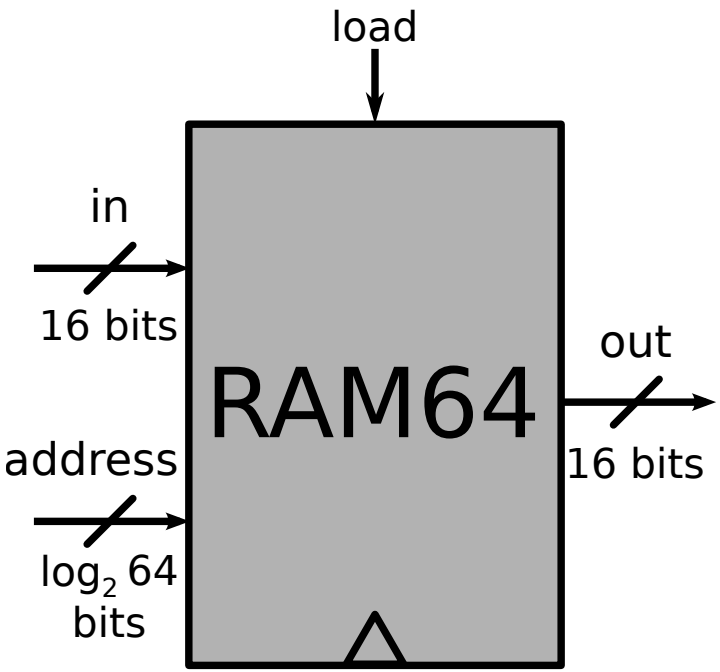

Figure: Input/Output ports of *circuit 2*, the RAM64 block.

Universiteit Utrecht

CPU block diagram

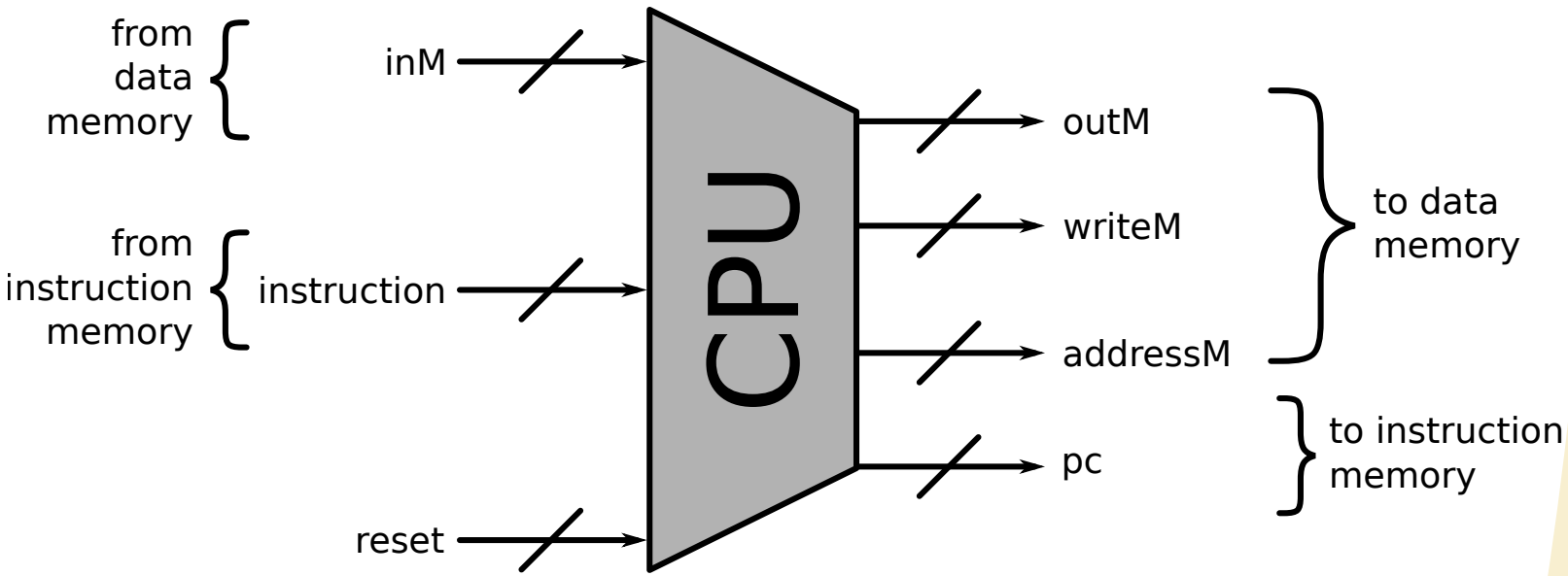

Figure: Input/Output ports of *circuit 3*, the *Hack* CPU.

Universiteit Utrecht

Section 4

Analysis of the EDSLs

Lava

ForSyDe

Universiteit Utrecht

Coquet

Universiteit Utrecht

Section 5

Conclusions

Thank you!

Questions?

Universiteit Utrecht