Gator: Toward a Correct Technology Mapper

A talk from your pal Andrew!!!

yeah!!!!!!

Hardware Compilation

First, what does a software compiler do?

```
// TODO @(bsaiki): andrew, do we need this?
int add_some_stuff() {
   a = 1;
   b = 2;
   c = 3;
   return (a + b) & c;
}
```

High-level **source code**

→

```
movq %rdi, $1
movq %rsi, $2
movq %rdx, $3
addq %rdi, %rsi
andq %rax, %rdx
```

Low-level implementation,
expressed as a series of **assembly instructions**

Hardware Compilation

What does a hardware compiler do?

```
module my_design
  (input a,
   input b,
   input c,
   output out);
  assign out = (a + b) & c;
endmodule
```

High-level **design fragment**

→

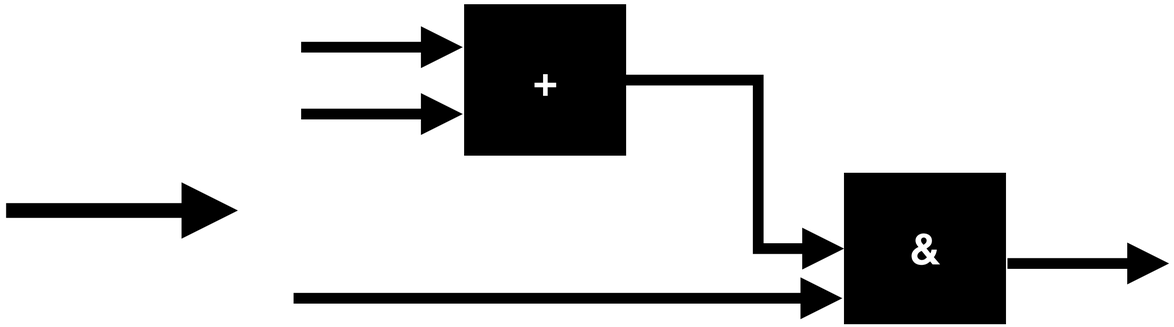
```
module my_design
  (input a,
   input b,
   input c,
   output out);
   wire mid;
   adder a0(.a = a, .b = b, .o = mid);
   ander a1(.a = mid, .b = c, .o = out);
endmodule
```

Low-level implementation,
expressed as
hardware primitives

Hardware Compilation

What does a hardware compiler do?

```
module my_design
  (input a,
   input b,
   input c,
   output out);
  assign out = (a + b) & c;
endmodule
```



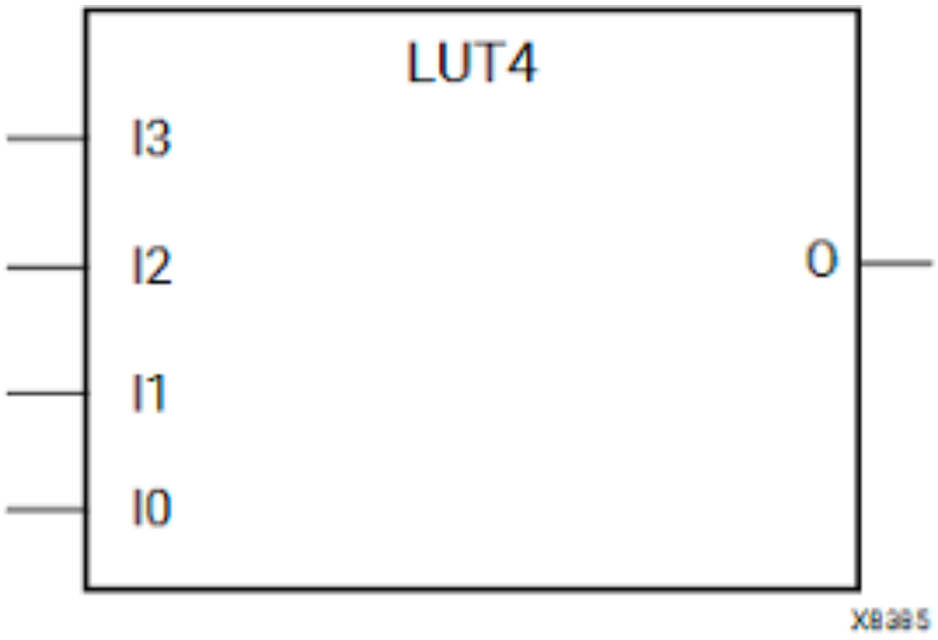
High-level **design fragment**

Low-level implementation,
expressed as
a picture on a rock

Primitives are complex!

LUT4

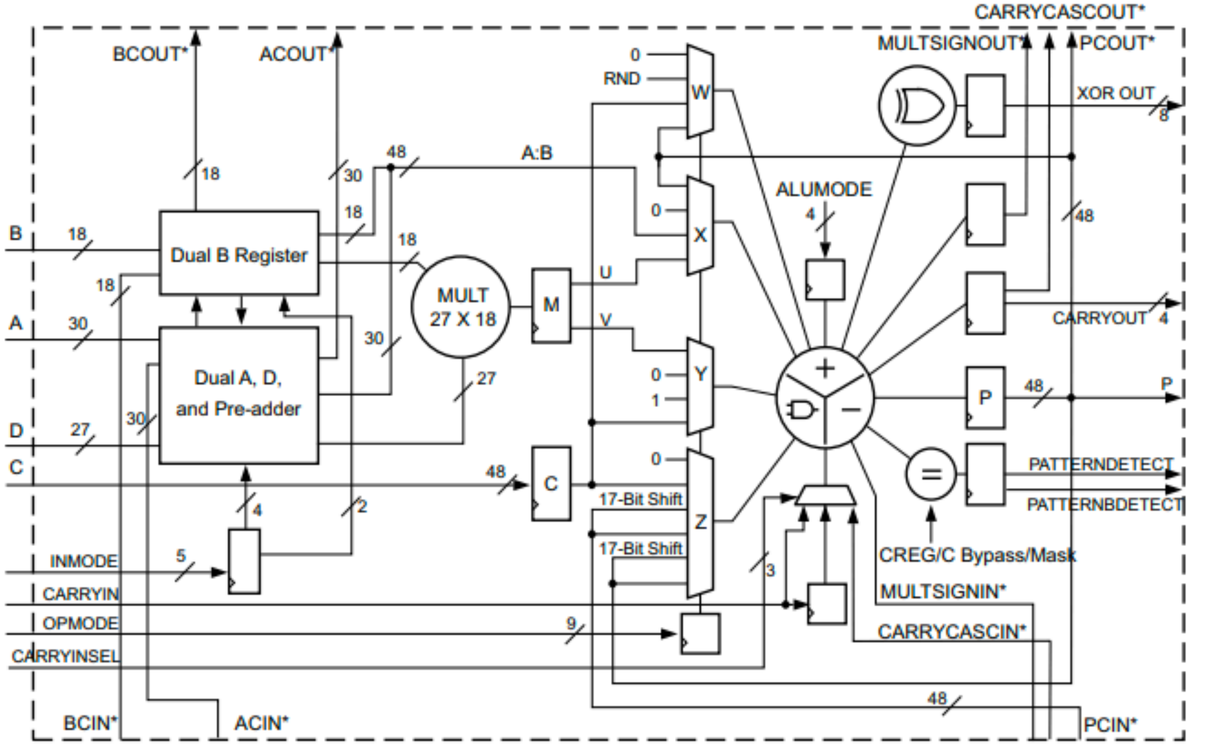
- 4-bit truth table
- Has state: internal memory



Primitives are complex!

DSP48E2

- Common use cases:
  - barrel shifting
  - SIMD operations
  - `(a + b) * c & d`
- Can be pipelined for 3 cycles



*These signals are dedicated routing paths internal to the DSP48E2 column. They are not accessible via general-purpose routing resources.

X16752-042617

Primitives are complex!

What can the DSP48E2 actually do?

increase in challenge for technology mappers [6]. The UltraScale+ DSP48E2 has over 100 ports and parameters, and the manual that explains how to configure them properly is over 75 pages long [10]. The large number of ports and parameters is due to the fact that these primitives

What does it mean for a primitive to be correct?

How do we know a DSP configuration is correct?

- If the DSP is configured correctly, then for some time t, it should:
  - compute the correct result on time t, which is true if
  - it had correct state on time (t - 1), which is true if:
  - it had correct state on time (t - 2), which is true if:
  - it had correct state on time (t - 3), which is true if:
  - ... shoot!

What does it mean for a primitive to be correct?

How do we know a DSP configuration is correct?

- If the DSP is configured correctly, then for some time t, it should:
  - compute the correct result on time t, which is true if
  - it had correct state on time (t - 1), which is true if:
  - it had correct state on time (t - 2), which is true if:
  - it had correct state on time (t - 3), which is true if:
  - ... shoot!

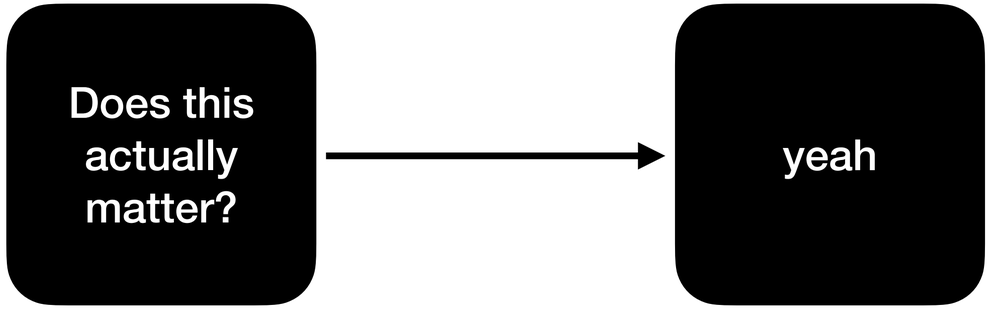
Solution from Vivado: don't provide these correctness guarantees!

Do correctness guarantees actually matter?

Verified Compilation in Software

Does this actually matter?

Verified Compilation in Software



Finding and Understanding Bugs in C Compilers

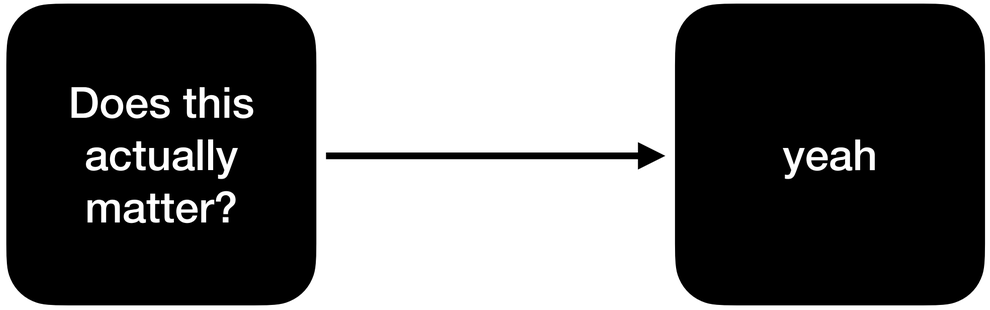
Xuejun Yang Yang Chen Eric Eide John Regehr
University of Utah, School of Computing
{jxyang, chenyang, eeide, regehr}@cs.utah.edu

Verified Compilation in Software

Verified Compilation in Hardware

Does this actually matter?

Verified Compilation in Hardware



Finding and Understanding Bugs in FPGA Synthesis Tools

Yann Herklotz
yann.herklotz15@imperial.ac.uk
Imperial College London
London, UK

John Wickerson
j.wickerson@imperial.ac.uk
Imperial College London
London, UK

Verified Compilation in Hardware

Formally Verified Hardware Compilation

Lutsig: A Verified Verilog Compiler for Verified Circuit Development

Andreas Lööw
Chalmers University of Technology
Gothenburg, Sweden

Formally Verified Hardware Compilation

Lutsig: A Verified Verilog Compiler for Verified Circuit Development

Andreas Lööw
Chalmers University of Technology
Gothenburg, Sweden

"Lutsig's technology mapped output netlists for this class of FPGAs contain <u>only k-LUT (with k ≤ 6) and carry4 cells</u>"

Formally Verified Hardware Compilation

Lutsig: A Verified Verilog Compiler for Verified Circuit Development

Andreas Lööw
Chalmers University of Technology
Gothenburg, Sweden

What about my DSP!? 🤬

CakeML / hardware

Type / to search

Code | Issues | Pull requests | Actions | Security | Insights

hardware

Public

Watch 42 | Fork 5 | Star 23

master | 5 Branches | 0 Tags | Go to file t | Add file | Code

AndreasLoow Manual merge of reviving word extract support, thanks @j4nk1 — 8264e60 · last year — 117 Commits

ag32	opentheory reader documentation	last year
compiler	handle indexing and slicing for internal inputs	last year
examples	More cleanup: newTranslator -> translator	3 years ago
misc	Reorganization (move around some directories etc.)	3 years ago
translator	Manual merge of reviving word extract support, thanks @j...	last year
verilog	handle indexing and slicing for internal inputs	last year
verilog_parser	Lutsig v2	3 years ago
.gitignore	Lutsig v2	3 years ago
.holpath	Add .holpath	6 years ago
LICENSE	BSD 3-clause license	6 years ago
README.md	More cleanup: newTranslator -> translator	3 years ago
hardwareMiscScript.sml	Lutsig v2	3 years ago
hardwarePreamble.sml	Lutsig v2	3 years ago
oracleScript.sml	Lutsig v1	4 years ago
sumExtraScript.sml	Lutsig v2	3 years ago
wordsExtraScript.sml	Split into multiple directories	6 years ago

README | BSD-3-Clause license

About

Verilog development and verification project for HOL4

hardware | verilog | synthesis | formal-methods | formal-verification | hol

Readme
BSD-3-Clause license
Activity
Custom properties
23 stars
42 watching
5 forks
Report repository

Contributors 3

AndreasLoow Andreas Lööw
xrchz Ramana Kumar
acjf3

Languages

Standard ML 94.4%
SystemVerilog 2.3%
Haskell 1.1%
Ruby 0.8%
Python 0.7%
Verilog 0.6%
Makefile 0.1%

CakeML / hardware

Type / to search

Code | Issues | Pull requests | Actions | Security | Insights

hardware Public

master | 5 Branches | 0 Tags | Go to file

AndreasLoow Manual merge of reviving word extract support, thanks @j4nk1

	opentheory reader documentation
	handle indexing and slicing for interna
	More cleanup: newTranslator -> trans
	Reorganization (move around some d
	Manual merge of reviving word extrac
	handle indexing and slicing for interna
	Lutsig v2
	Lutsig v2
	Add .holpath
	BSD 3-clause license
README.md	More cleanup: newTranslator -> trans
hardwareMiscScript.sml	Lutsig v2
hardwarePreamble.sml	Lutsig v2
oracleScript.sml	Lutsig v1
sumExtraScript.sml	Lutsig v2
wordsExtraScript.sml	Split into multiple directories 6 years ago

SystemVerilog 2.3% Haskell 1.1%
Ruby 0.8% Python 0.7%
Verilog 0.6% Makefile 0.1%

README BSD-3-Clause license

CakeML / hardware

Type / to search

Code | Issues | Pull requests | Actions | Security | Insights

hardware Public

Watch 42 | Fork 5 | Star 23

master

Andreasl

ag32
compiler
examples
misc
translator
verilog
verilog_pa
.gitignore
.holpath
LICENSE
README.m
hardwareM
hardwareP
oracleScri

sumExtraScript.sml	Lutsig v2	3 years ago
wordsExtraScript.sml	Split into multiple directories	6 years ago

and verification

synthesis
nal-verification hol

nse

ndreas Lööw

umar

Standard ML 94.4%
SystemVerilog 2.3% Haskell 1.1%
Ruby 0.8% Python 0.7%
Verilog 0.6% Makefile 0.1%

README BSD-3-Clause license

If a hardware compiler is **extensible,** support for additional primitives can be added with little user effort.

If a hardware compiler is **extensible,** support for additional primitives can be added with little user effort.

In general, hardware compilers aren't extensible!

Lakeroad: an extensible compilation tool

- Lakeroad uses **program synthesis** to map high-level designs to low-level **hardware primitives**.
- Lakeroad reasons about what a primitive can do through **automatic extraction of SMT semantics** from **vendor-provided simulation models.**

Lakeroad: an extensible compilation tool

- Lakeroad uses **program synthesis** to map high-level designs to low-level **hardware primitives**.
- Lakeroad reasons about what a primitive can do through **automatic extraction of SMT semantics** from **vendor-provided simulation models**.
- If someone wants to support a new DSP, they've got to:

Lakeroad: an extensible compilation tool

- Lakeroad uses **program synthesis** to map high-level designs to low-level **hardware primitives**.
- Lakeroad reasons about what a primitive can do through **automatic extraction of SMT semantics** from **vendor-provided simulation models**.
- If someone wants to support a new DSP, they've got to:
  - Download the simulation model

Lakeroad: an extensible compilation tool

- Lakeroad uses **program synthesis** to map high-level designs to low-level **hardware primitives**.
- Lakeroad reasons about what a primitive can do through **automatic extraction of SMT semantics** from **vendor-provided simulation models**.
- If someone wants to support a new DSP, they've got to:
  - Download the simulation model
  - Write a short sketch

Lakeroad: an extensible compilation tool

- Lakeroad uses **program synthesis** to map high-level designs to low-level **hardware primitives**.
- Lakeroad reasons about what a primitive can do through **automatic extraction of SMT semantics** from **vendor-provided simulation models**.
- What does the workflow of Lakeroad actually look like?

Lakeroad's Compilation Flow

```
(a + b)
```

ALU

1. Download the simulation model

```
(a + b)
```

ALU

ALU.v

1. Download the simulation model

```
(a + b)
```

ALU

I can add and multiply!

ALU.v

1. Download the simulation model

```
(a + b)
```

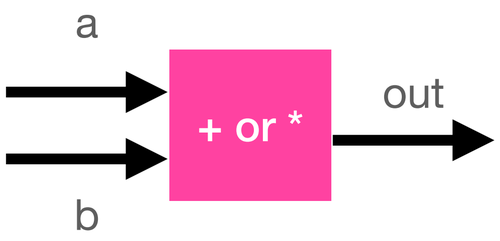
+ or *

I can add and multiply!

ALU.v

2. Set up a sketch

```
(a + b)
```



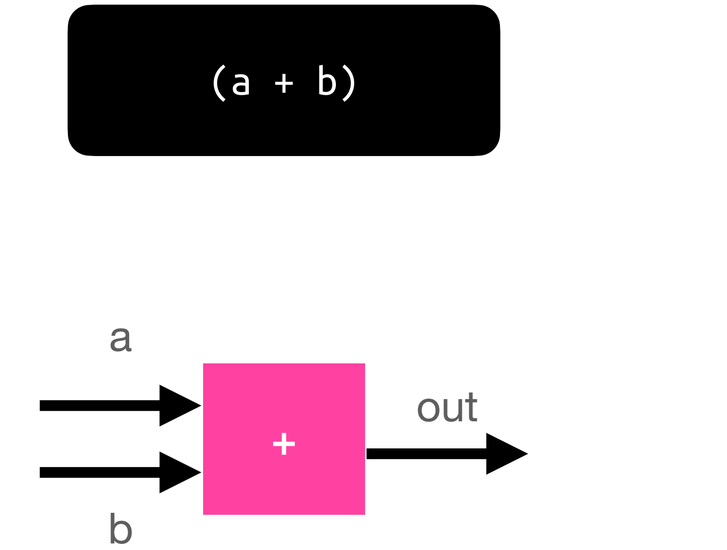
ALU.v

3. Lakeroad's synthesis query:

```
def impl(a, b, t):
   return a + b

def sketch(a, b, t):
  return a (+ or *) b

assert (forall a, b,
  impl(a, b, 1) == sketch(a, b, 1) and
  impl(a, b, 2) == sketch(a, b, 2)
```



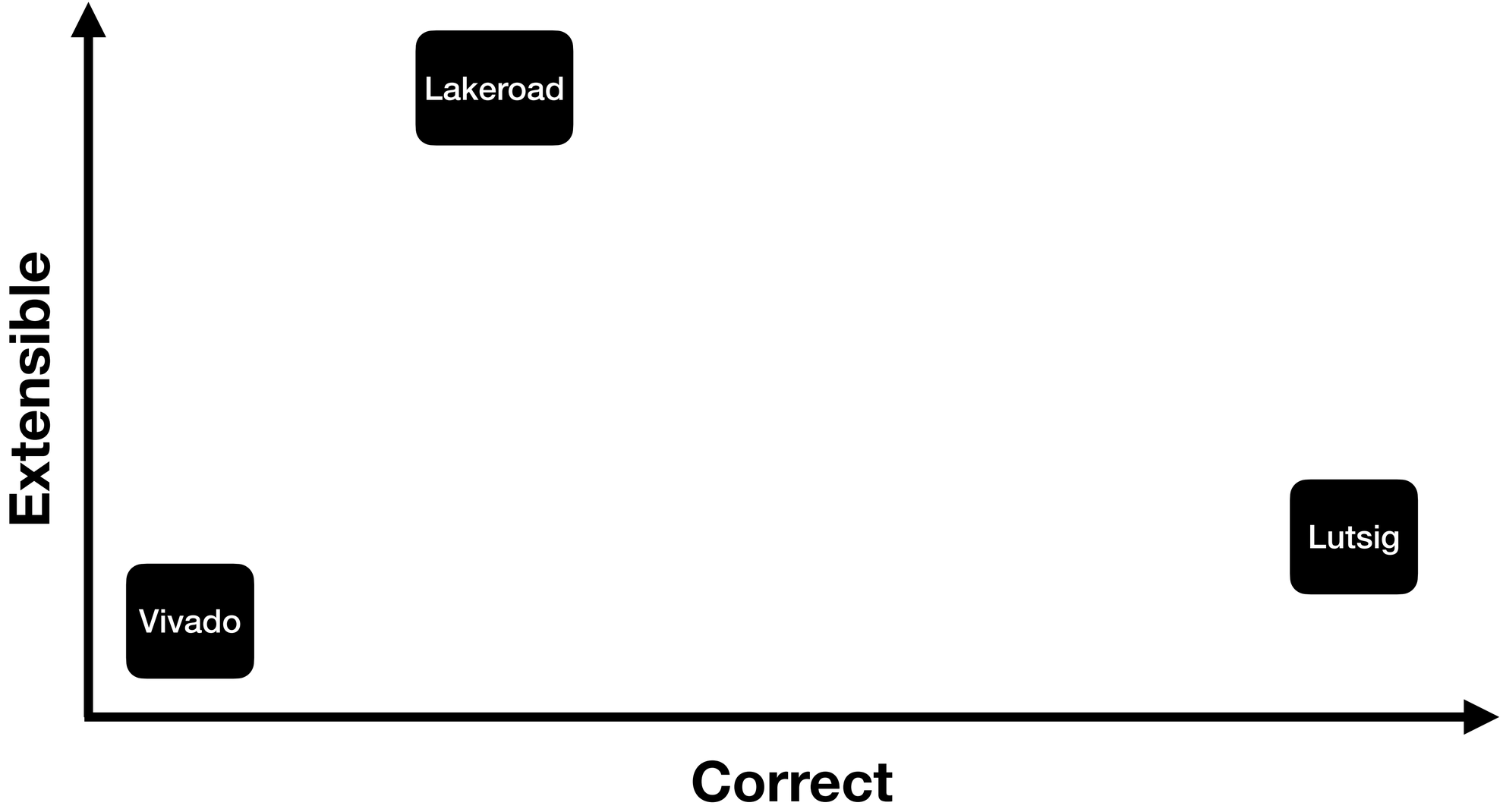
ALU.v

Lakeroad's correctness guarantees

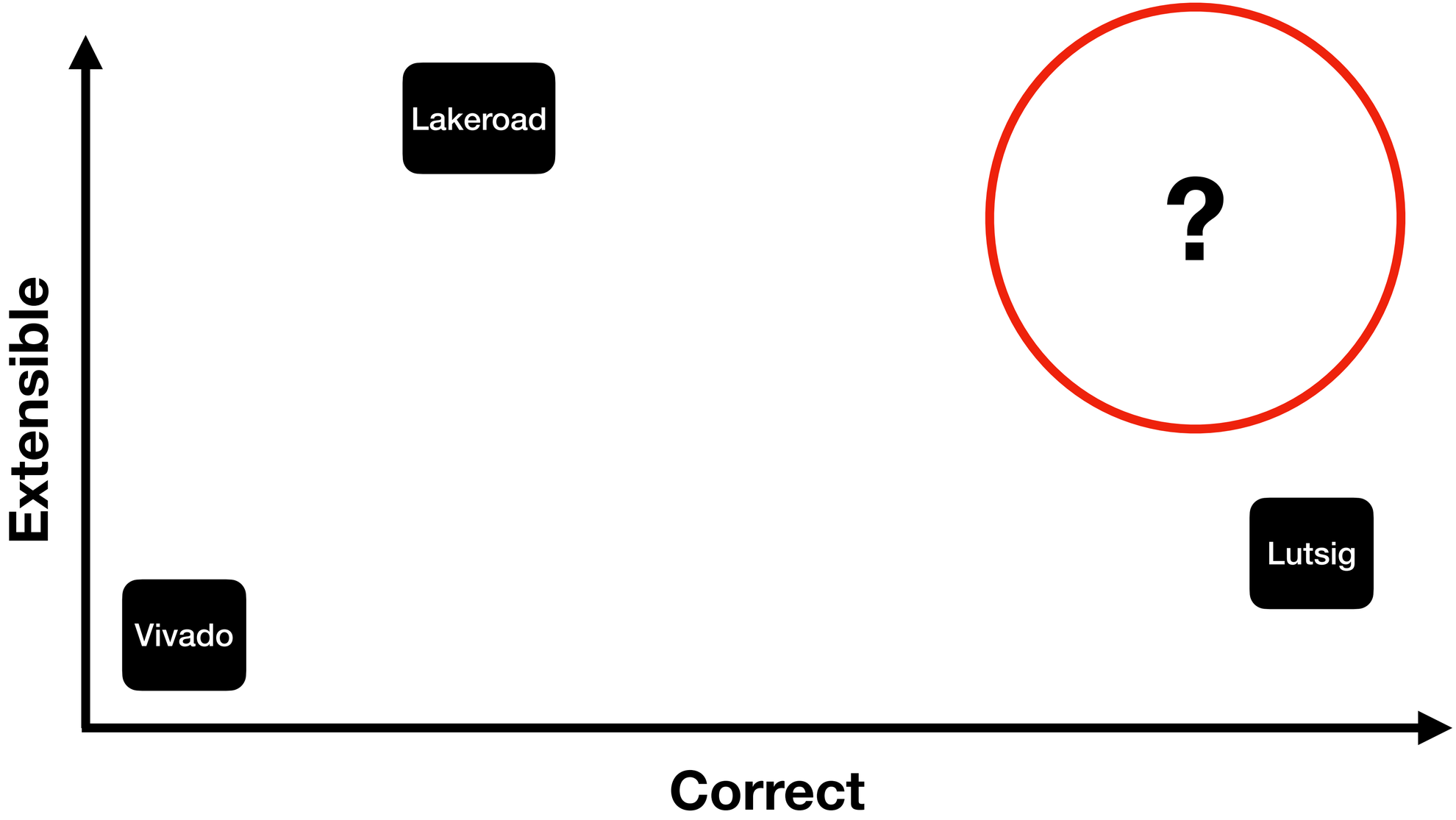
- Lakeroad's program synthesis query does a **"bounded model synthesis"** where correctness for the first few cycles is formally guaranteed.
- ...but this doesn't account for all the other cycles!
- Lakeroad provides **some guarantees** for correctness, but not **full** guarantees.

```
assert (forall a, b,
  impl(a, b, 1) == sketch(a, b, 1) and
  impl(a, b, 2) == sketch(a, b, 2)
```

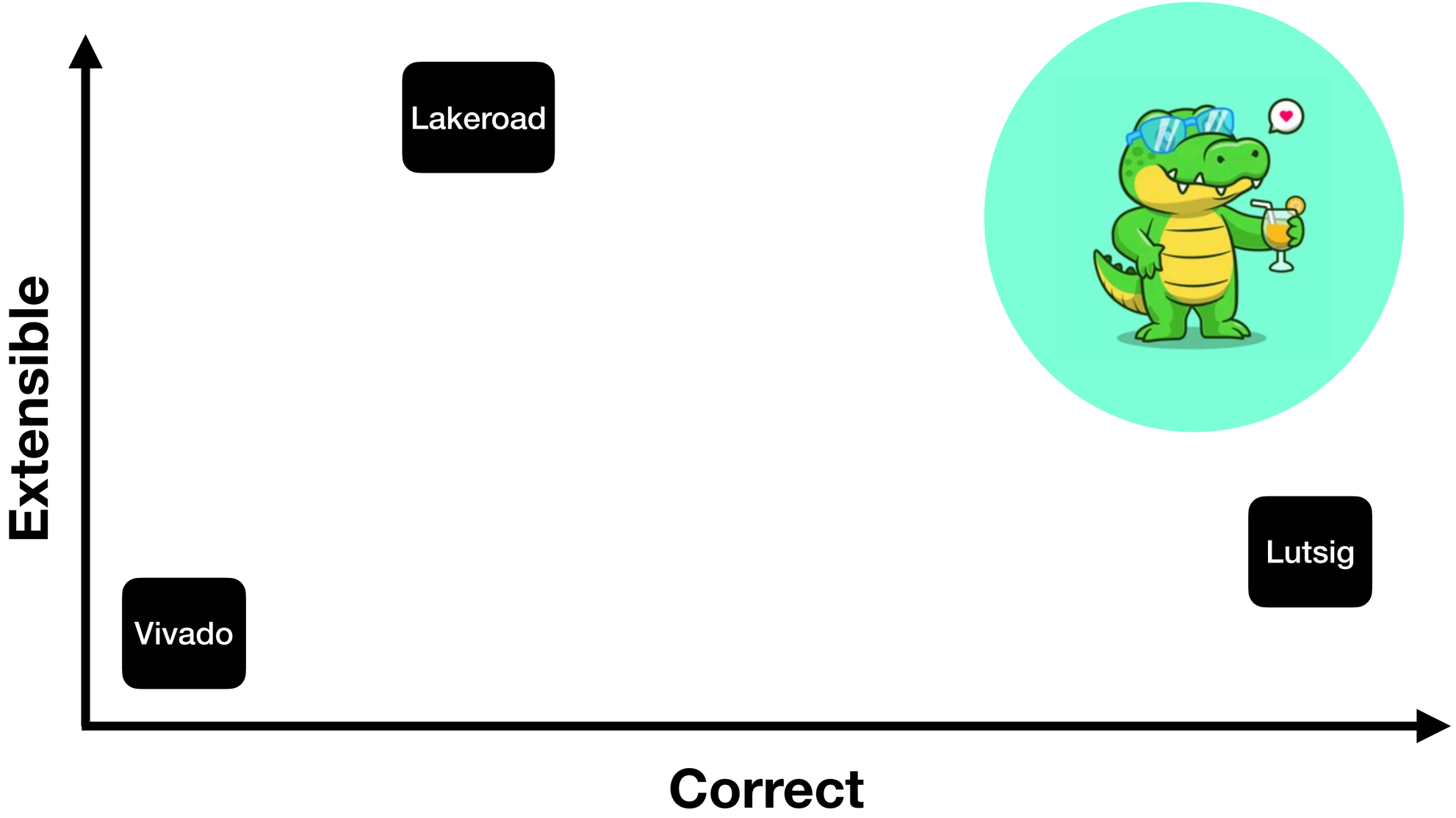
A Survey of Hardware Compilers



A Survey of Hardware Compilers



A Survey of Hardware Compilers



🐊 The Gator™ Project™ ©:

- Goal: a hardware compiler which is **correct** and **extensible.**
- What if we modify Lakeroad's synthesis query so that it's correct **for all time?**

$$\forall i : spec(i,1) = impl(i,1) \wedge spec(i,2) = impl(i,2)$$

$$\Downarrow$$

$$\forall i, t : t > init \rightarrow spec(i,t) = impl(i,t)$$

Demo Time! (maybe)

Thank you!!