Safer and Faster Kami

Swarn Priya and Murali Vijayaraghavan
(Special thanks to Puck)

Introduction

- A **Coq-based DSL** for writing **hardware** designs.
- **Hardware Block in Kami**

- **Trace in Kami :**
  Sequence of labels corresponding to the sequence of rule execution.
  {(RegsT, Rule/Method, MethsT), ……..}

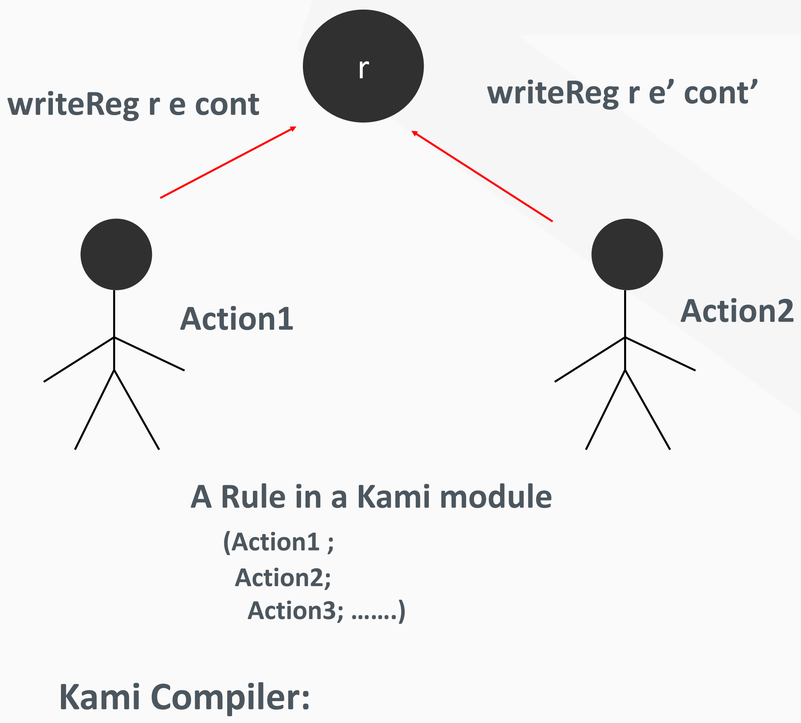
Kami is more safe now!

How?

- Kami programs failed at run-time due to double-writes.
- Now, Kami supports check for double-writes at the compile time.

What is double-writes in Kami?

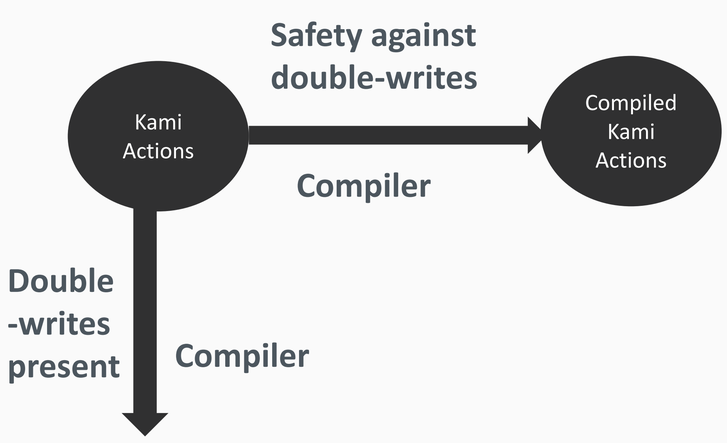
- When a register is being written more than once within a rule.
- What is a rule in Kami?
  Bunch of actions (read, write, meth …) executing one after the other.



A Rule in a Kami module

(Action1 ;
Action2;
Action3; …….)

Kami Compiler:



Error! Go back and fix it

Support for New Word Library

New Word Library which aims in making the life of Kami coders easy.

Purpose:

- Help in performing operations on bits.
- Representation of bits in Z.

Why?

- Old library was tedious to use.
- It is outdated and not maintained.
- Hard to understand and use for non-Kami coders.

How?

- Involving Z to represent bits.
- Z is a well supported library in Coq which give us privilege to use the existing supported theorems.
- More easy to work with as we need to think in terms of numbers than bits.
- Proofs involving new word representation is easier.
- All the operations involve either arithmetical operations or logical operations while in the old library operations mostly involve recursion.
- Development of proofs is faster as compared to the old library.

Old Word Notation:

```
Inductive word : nat -> Set :=
| WO : word 0
| WS : bool -> forall n, word n -> word ( S n).
```

New Word Notation:

```
Record word :=
   mk {wordVal : Z;
       wordBound : wrap_value wordVal = wordVal}.

Local Notation wrap_value n :=
   (Z.modulo n (Z.pow 2 (Z.of_nat width))).
```

Operations supported by the library are:

- Addition, Subtraction, Multiplication, Division, Modulo, OR, AND, XOR, 1's compliment, 2's compliment, Unary And, Unary OR, Unary XOR Left Shift, Right Shift

These operations in the new library involves arithmetic and logical operations only.

Kami Tutorial

Goal:

- Tutorial to broadcast the idea whether Kami design (**actual implementation**) implements another simpler designs (**specification**).
- An outsider need not to know about the details about the semantics of Kami but could use the tactics to prove that implementation adhere to their specification.

Theorem-prover based proof

Example:
Implementation : One Element FIFO
Specification: Two Element FIFO Array

One Element FIFO ⊑ Two Element FIFO Array

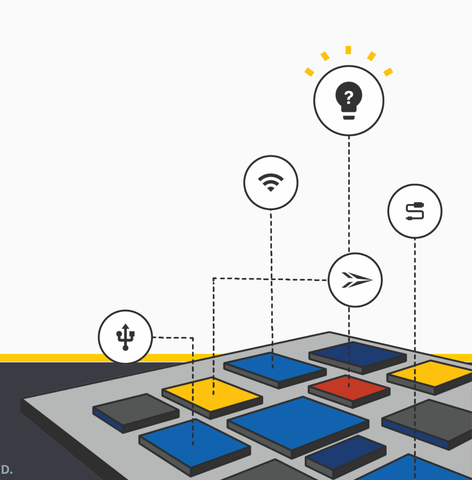
Dequeue/Enqueue Dequeue/Enqueue

This relation is satisfied in presence of bunch of invariants.

Formally verified theorem:
Any trace that can be produced by One Element FIFO can also be produced by Two Element FIFO Array.

Conclusion

- Safety against double-writes at compile time might help us to get rid of serious failures at runtime.
- New Word Library is more user friendly
- Tutorials help outsiders (non-Kami coders) to get an overview of Kami as well as use it for formally verifying their own modules.



PURDUE UNIVERSITY
Swarn Priya
PhD Student, Purdue University
priyas@purdue.edu

Murali Vijayaraghavan
Program Manager, SiFive Inc.
vmurali@sifive.com