Small Example on rewrite rules

Thanawat Techaumnuaiwit

July 23, 2023

Install

- My github repo is on: https://github.com/thiskappaisgrey/research-stuff
- If you want to build it, you can use *nix* to install the dependencies.
- You do need the forked repo of lakeroad-yosys (I failed to do git submodules correctly, so for now, you'll have to clone it yourself) with the egglog branch for this to work..
- The nix derivation already includes teh yosys dependencies to build.

Half Adder example

- The simplest example is finding half adders in a full-adder implementation
- We can use egglog to do *subgraph isomorphism* - find the half-adder in the full adder!
- Basically - we build the component as a subgraph and use Egglog to find that subgraph in a bigger graph. This example only has the 1 "naive" rewrite rule.
- Run `just build-full-adder` to build the example.

Half adder

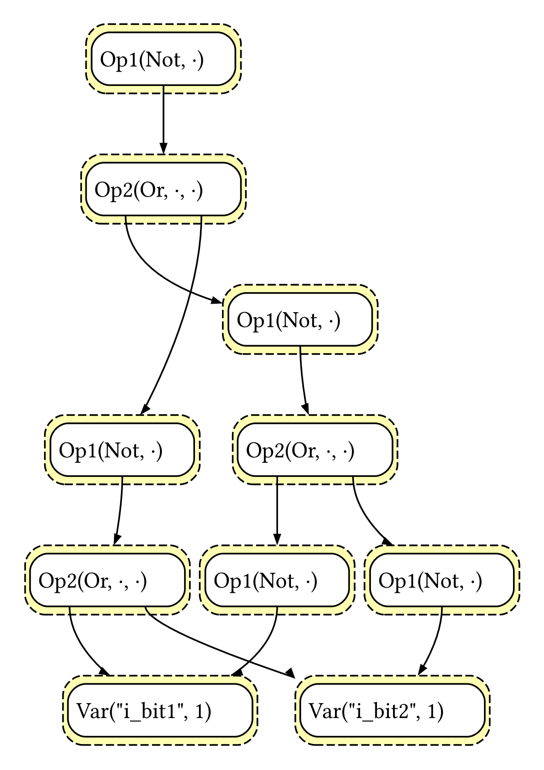

Figure 1: The half adder is used to build up the rewrite rule

Full Adder

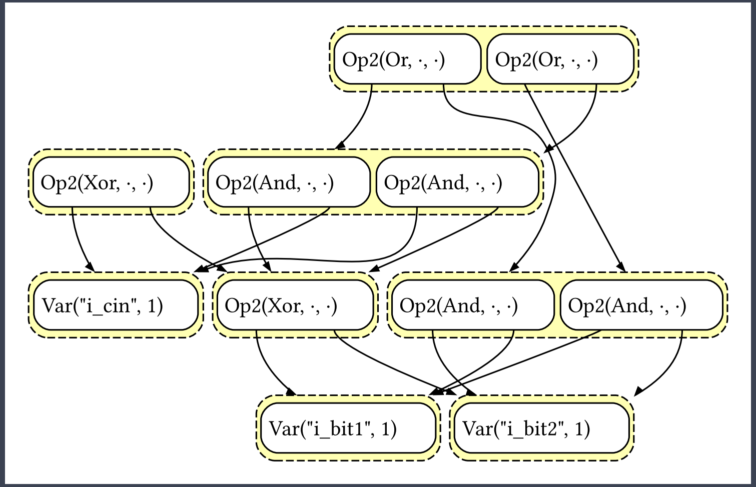

Figure 2: The rewrite rule built by the half adder is used to search for it in the full adder

XOR example

- Finding an XOR gate in a graph with the XORs compiled away(i.e. translated to and/not gates) is much harder.
  - Nodes can be "merged" together during the compilation process where the XORs can be harder to find
- We can't just use "subgraph isomorphism" to find the XOR implementation anymore! We "continuosly deform" the graph using more rewrite rules in order to find the XOR implementation!
- I'll call this problem - "subgraph homotopy"[1]

[1] I'm not a mathematician but I'm using big words to sound smart

XOR compiled

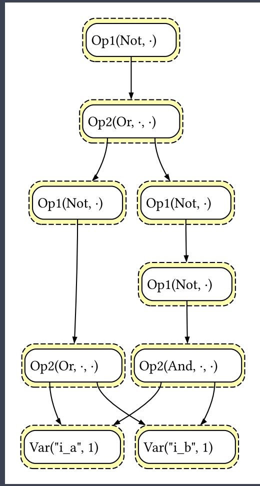

Figure 3: This is the XOR compiled

Rewrites that were added

```
;; absorbtion.
(rewrite
 (Op2 (Or) x (Op2 (And) x y))
 x
 :ruleset deoptimize)

(rewrite
 (Op2 (And) x (Op2 (Or) x y))
 x
 :ruleset deoptimize)
;; commutativity
(birewrite
```

```
 (Op2 (And) x y)
 (Op2 (And) y x)
 :ruleset deoptimize)
(birewrite
 (Op2 (Or) x y)
 (Op2 (Or) y x)
 :ruleset deoptimize)

;; idempotence
(rewrite
        (Op2 (And) x x)
        x
        :ruleset deoptimize
```

```
  )
(rewrite
        (Op2 (Or) x x)
        x
        :ruleset deoptimize
  )
;; De Morgansk

(birewrite
        (Op1 (Not) (Op2 (And) x y))
        (Op2 (Or) (Op1 (Not) x) (Op1 (Not) y))
        :ruleset deoptimize
  )
```

```
(birewrite
        (Op1 (Not) (Op2 (Or) x y))
        (Op2 (And) (Op1 (Not) x) (Op1 (Not) y))
        :ruleset deoptimize
  )
```

Looking for XORs in a “bigger” circuit

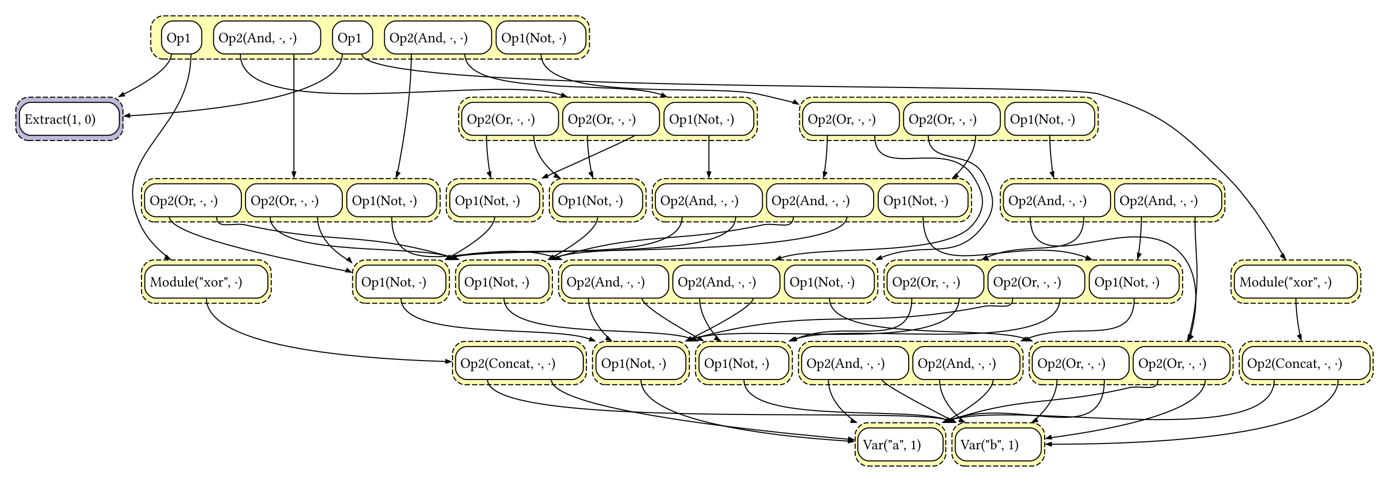

Figure 4: Here, we find the compiled XOR in a slightly bigger circuit by adding the above rewrites

The “circuit”

```
module test(
        input a,
        input b,
        output d,
        output c
);
        assign d = a & b;
        assign c = a ^ b;
endmodule
```

Conclusion

- We can use a number of passes to find our component in a bigger circuit.. i.e. continuosly deform the graph into the form that we "want" to find.
- Some difficulties after we find the module is *extracting*. ENodes can be a part of multiple modules - we have to ensure that they are "comsumed" by the modules at most once.