How do I use all this?

Outline

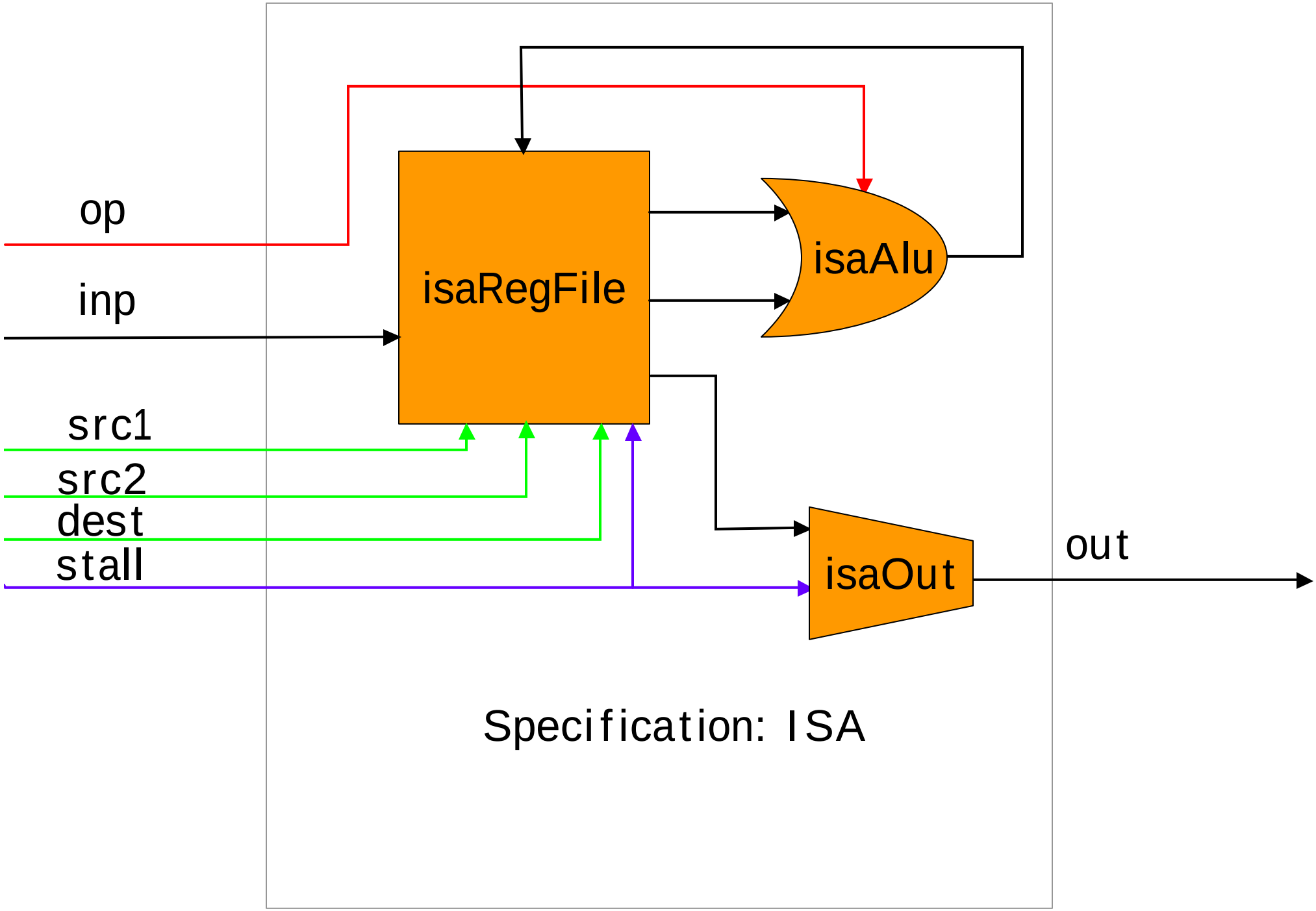
1 Informal Introduction

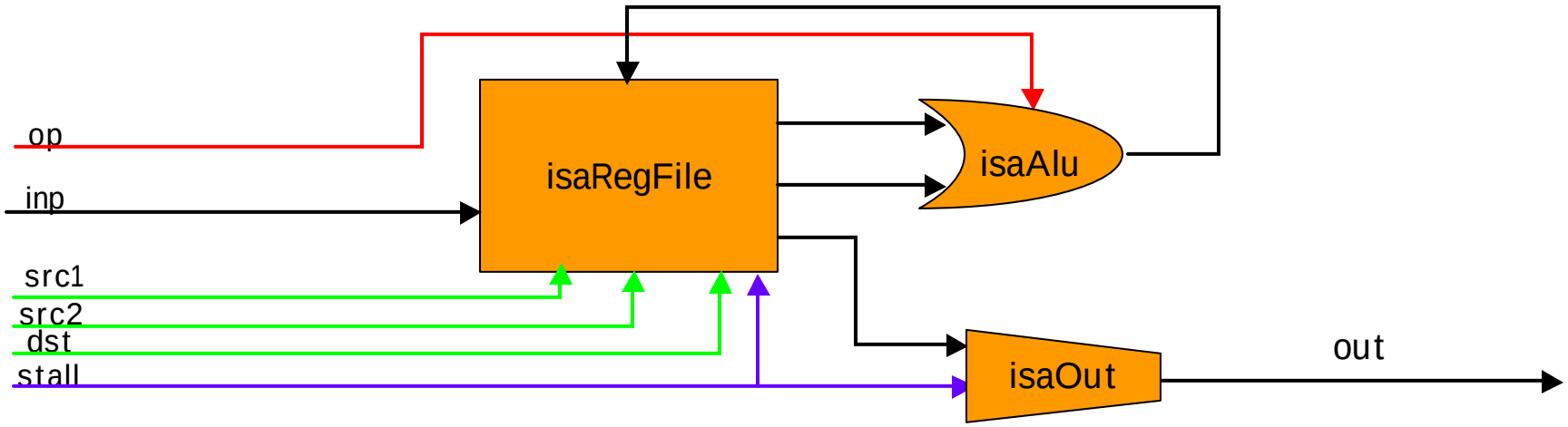
2 Formal Definitions

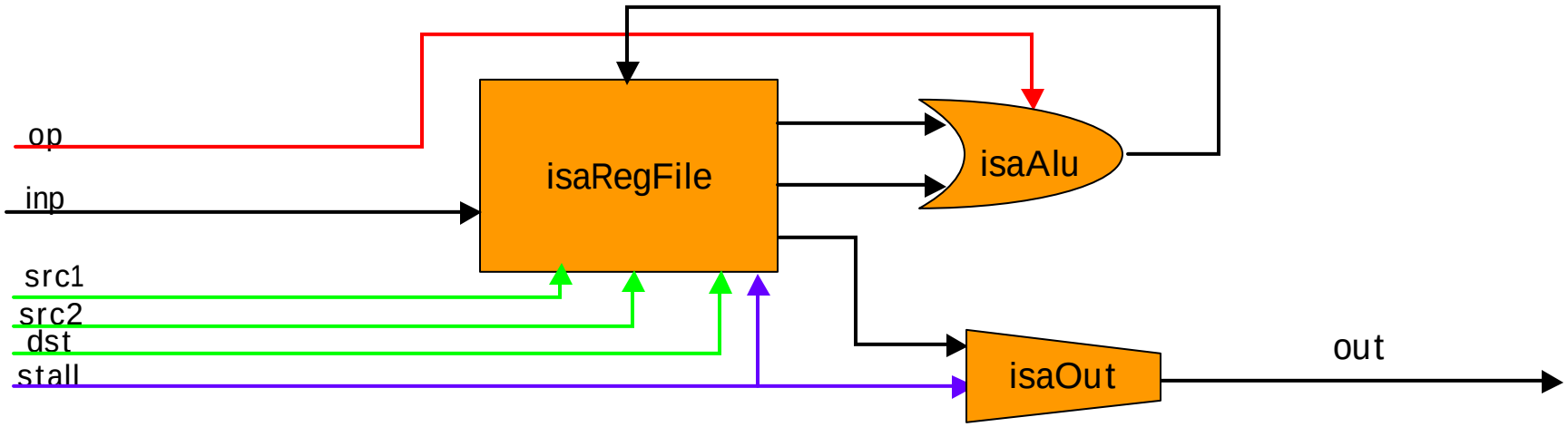
- Reactive Systems
- Witnessed Refinement Proofs
- Slicing Reactive Systems
- Decomposing Refinement Proofs

3 Formal Example: Three-Stage Pipeline

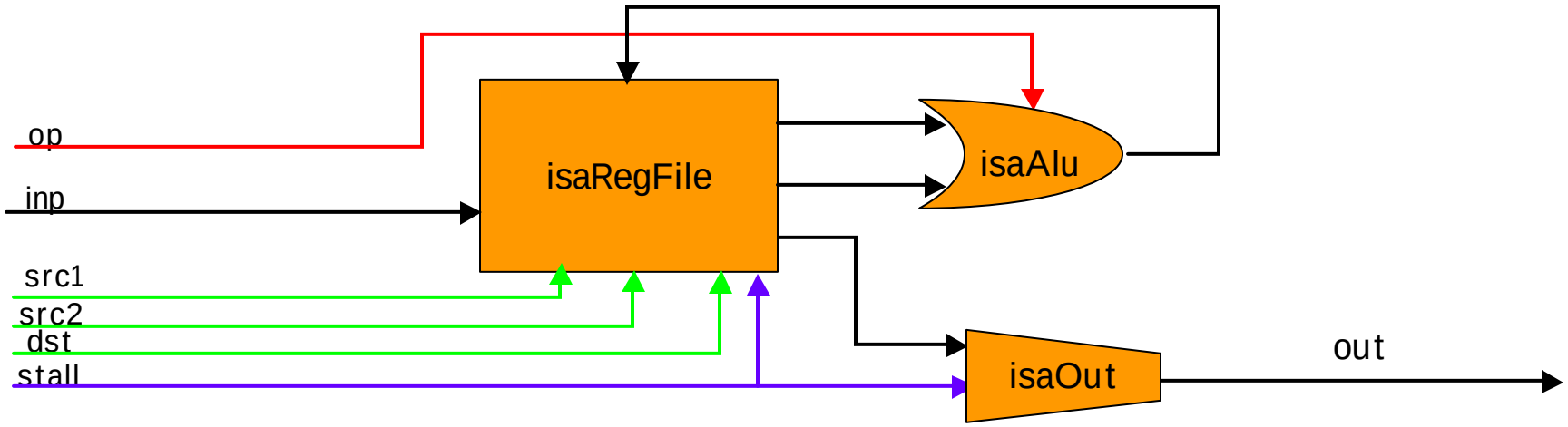
4 Informal Example: Dataflow Processor Array


op
inp
src1
src2
dest
stall
isaRegFile
isaAlu
isaOut
out
Specification: ISA

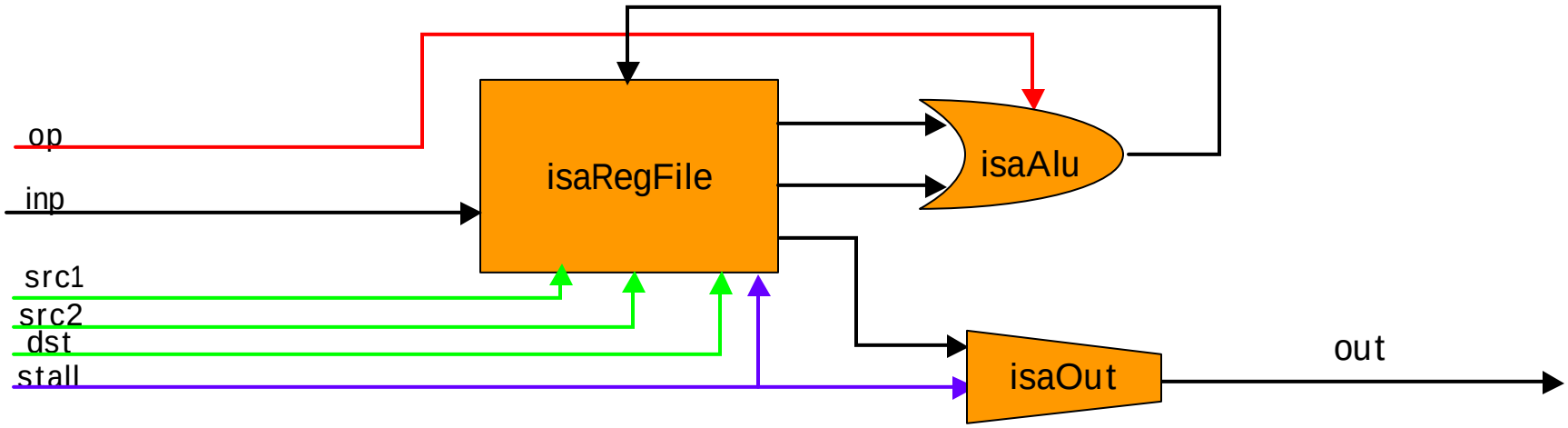

op
inp
src1
src2
dst
stall
isaRegFile
isaAlu
isaOut
out




```
load    r1  1
xnor    r2  r1  r1
store   r2
```



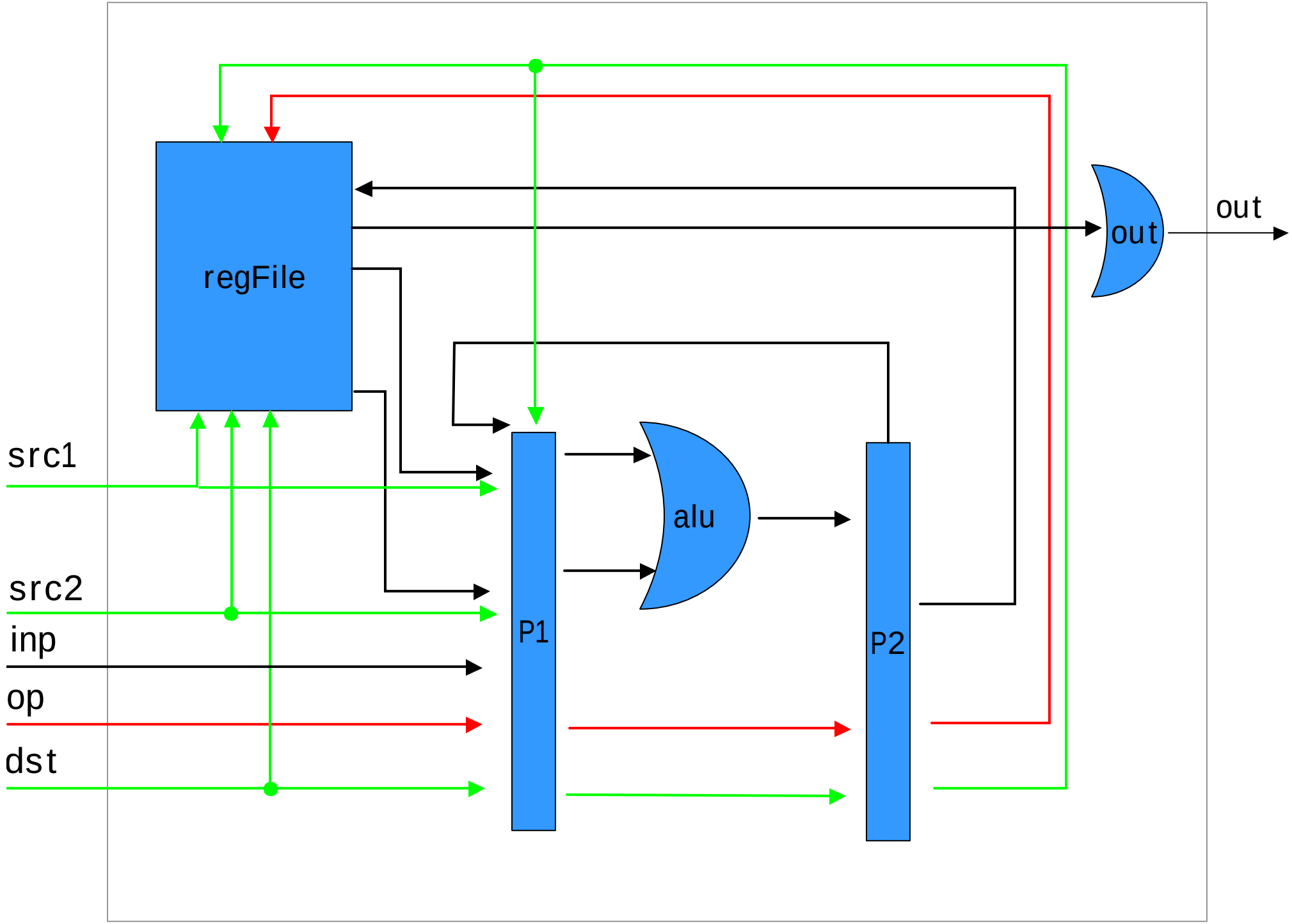
load	r1	1		r1	:=	1
xnor	r2	r1	r1	r2	:=	0
store	r2			out	:=	0

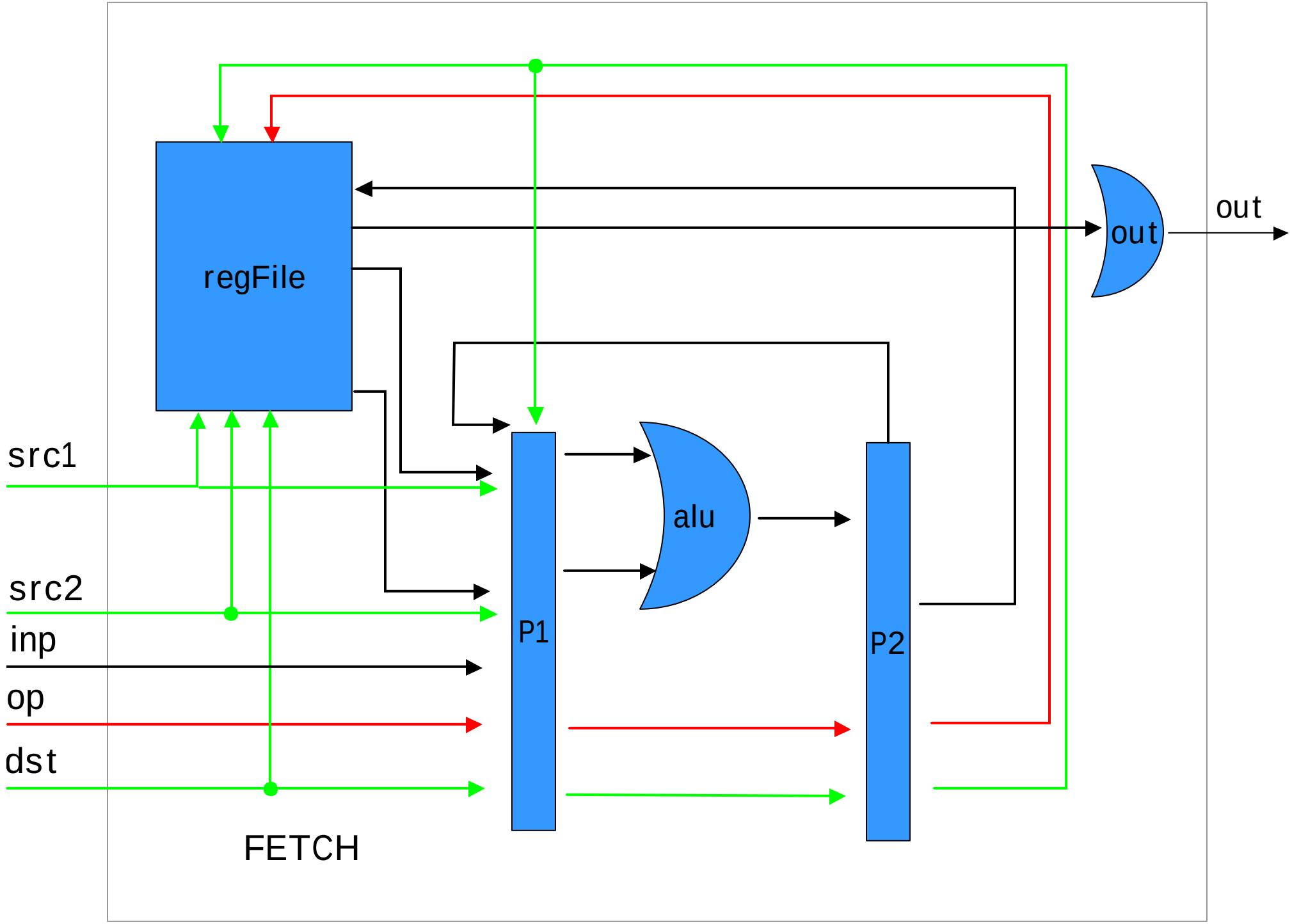


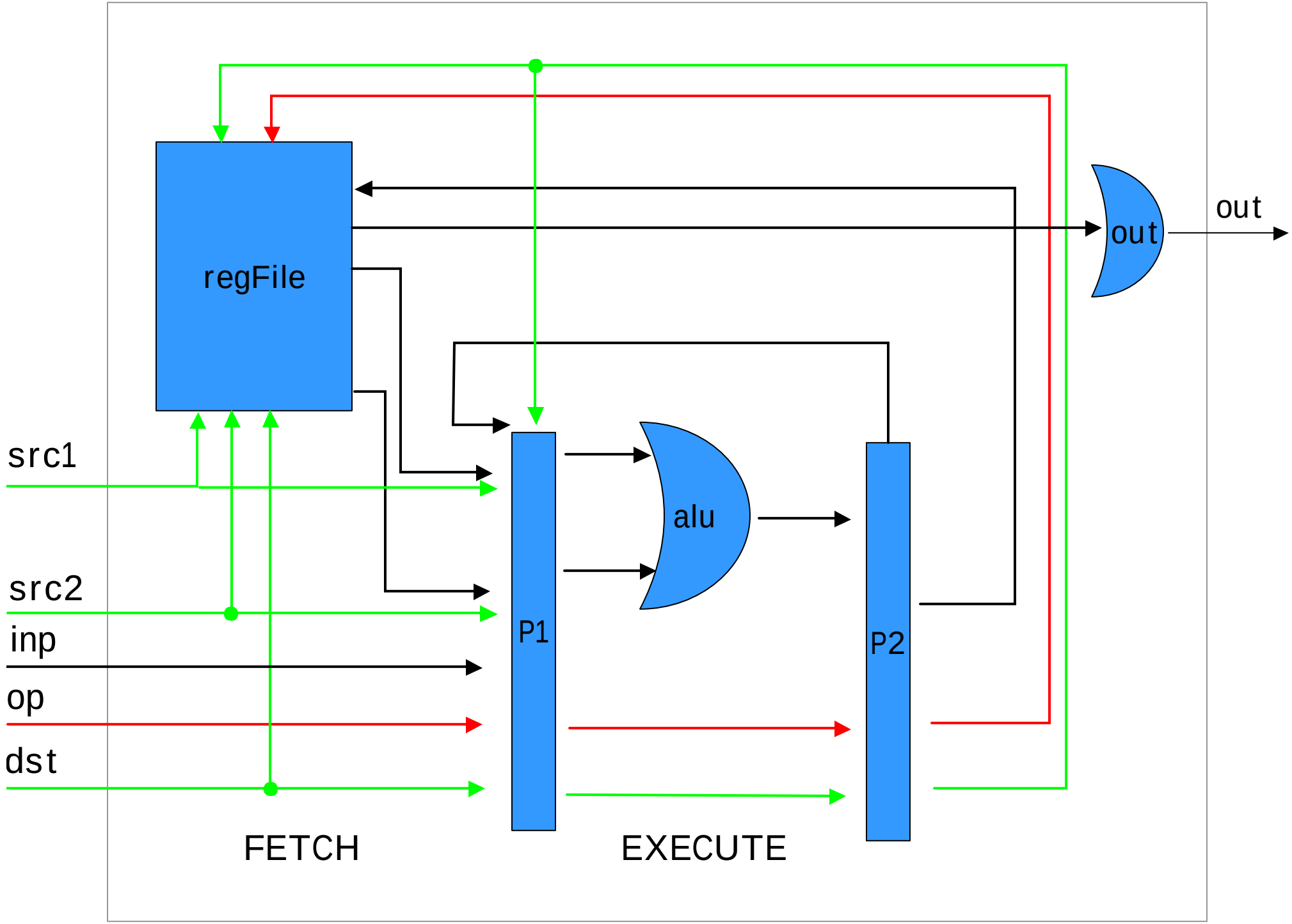
load	r1	1		r1 :=	1
xnor	r2	r1	r1	r2 :=	0
store	r2			out :=	0

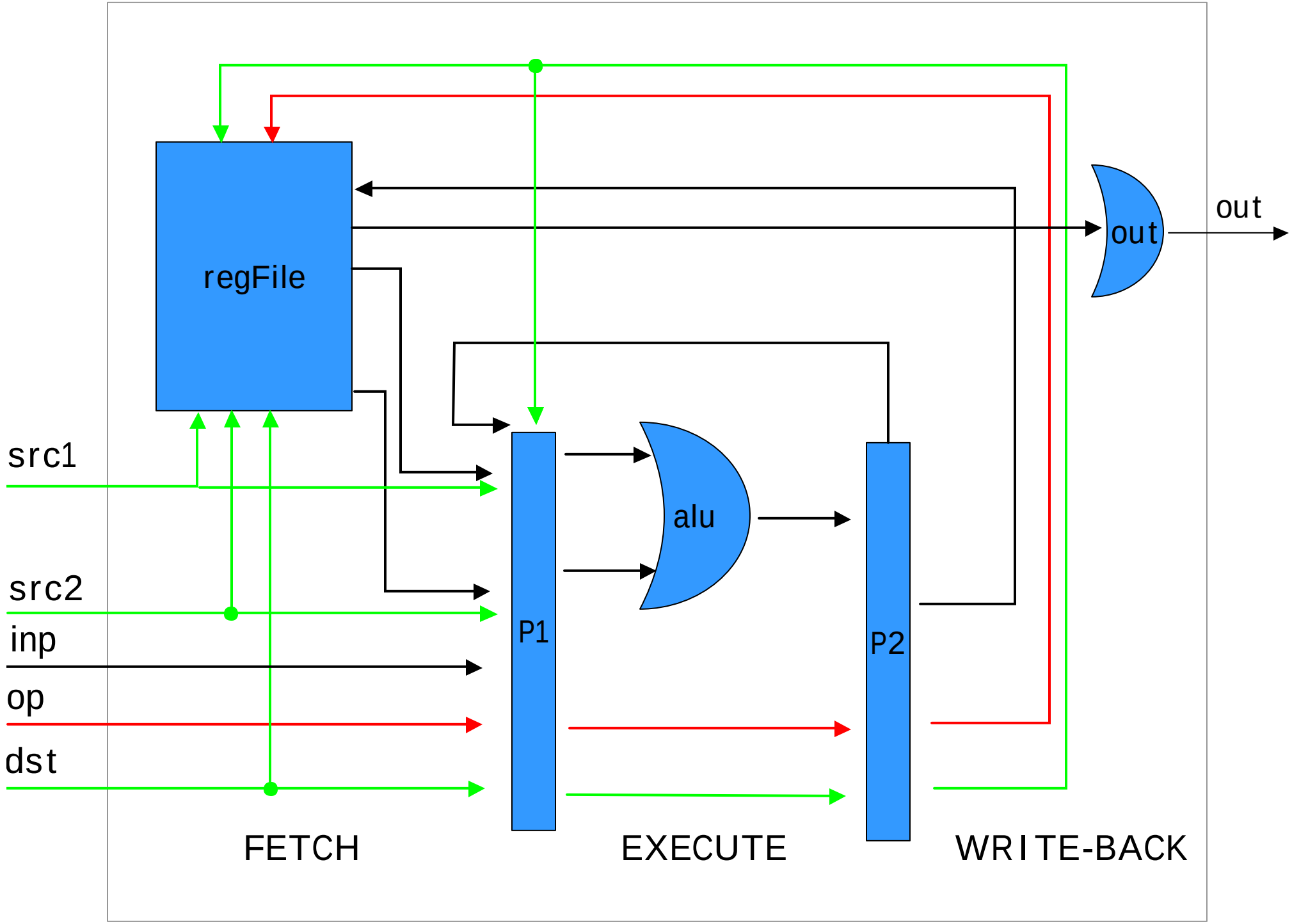
Notes:

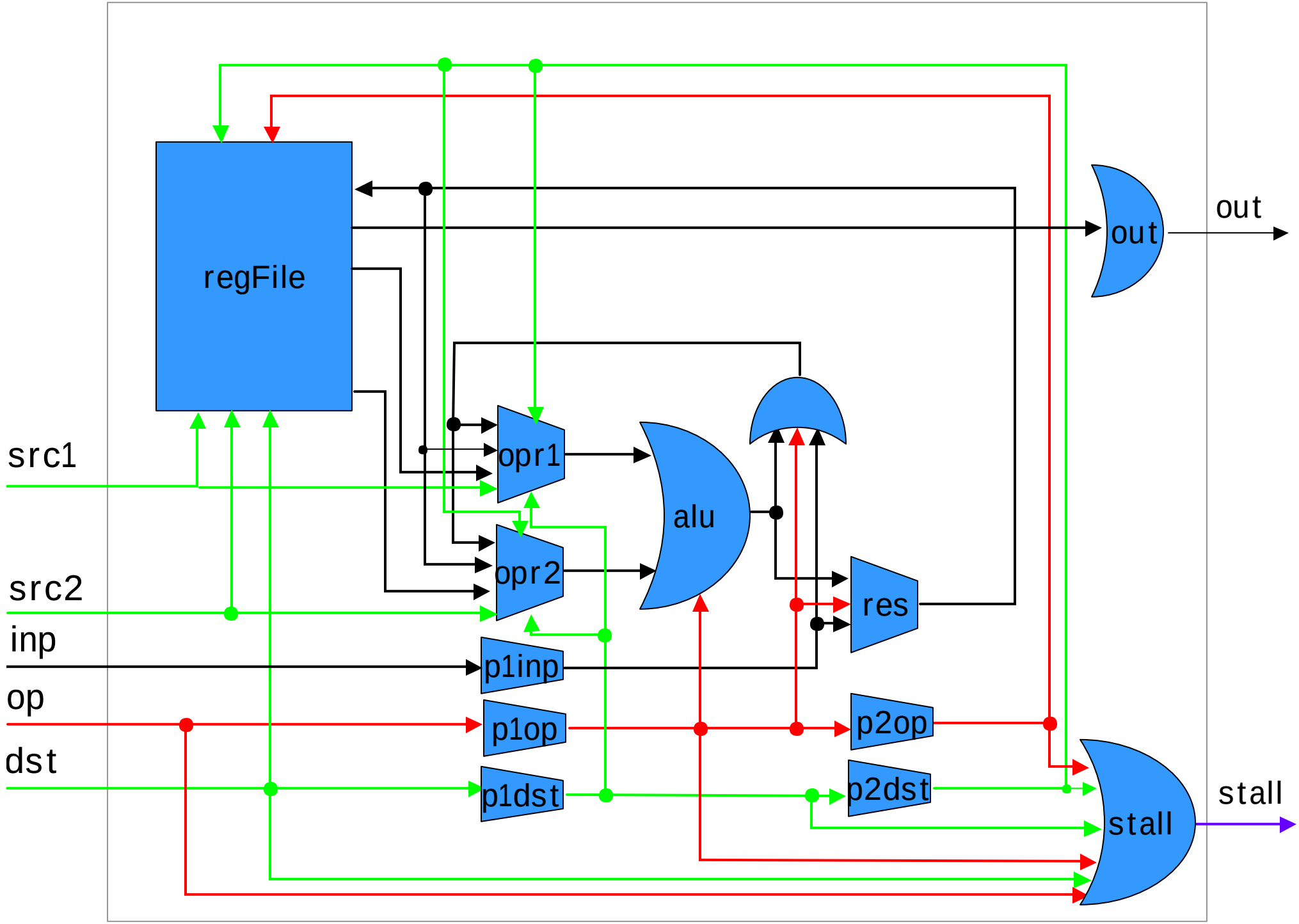
1. Store instruction results in an output
2. Memory hierarchy is not represented
3. Why do we need "stall" ?

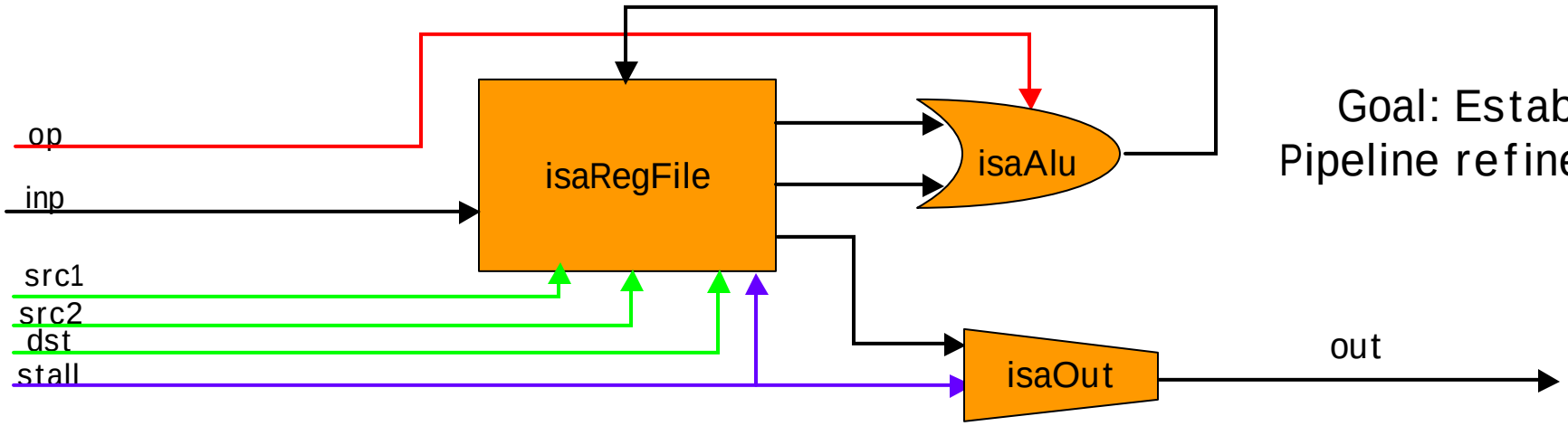

regFile
out
out
alu
P1
P2
src1
src2
inp
op
dst


regFile
out
out
alu
P1
P2
src1
src2
inp
op
dst
FETCH

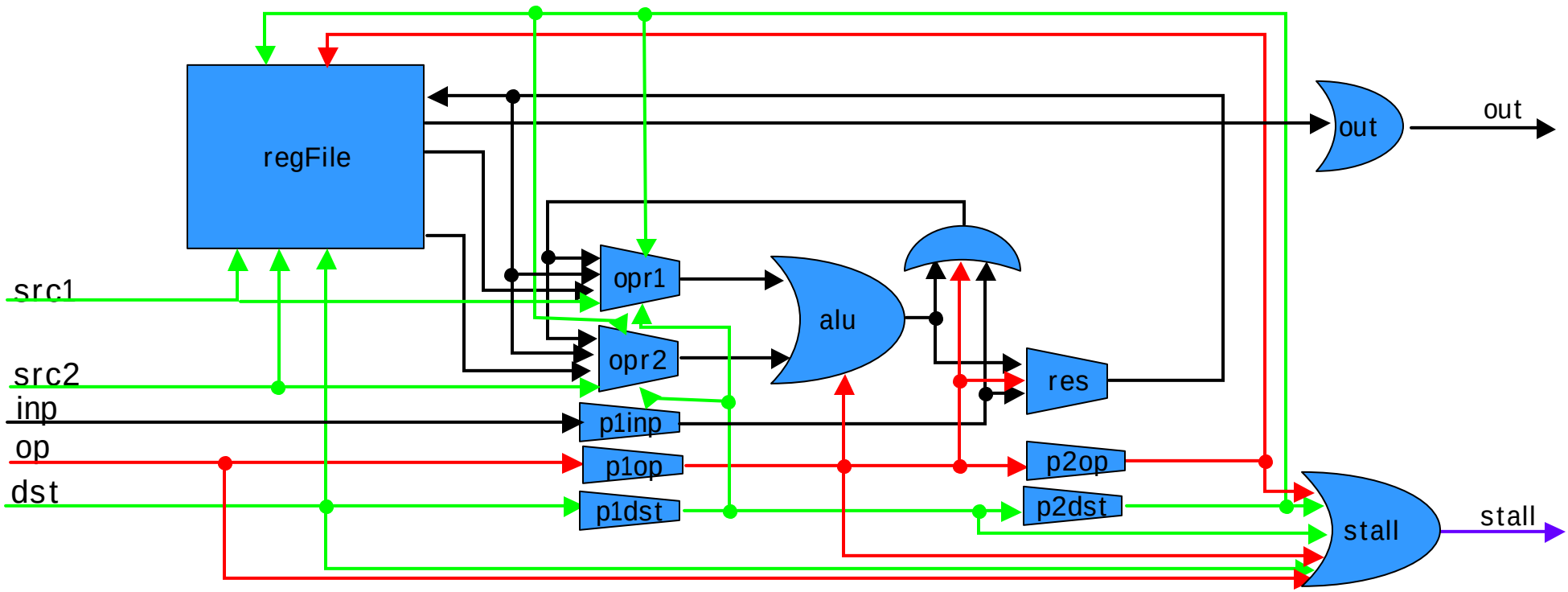

regFile
out
out
alu
P1
P2
src1
src2
inp
op
dst
FETCH
EXECUTE

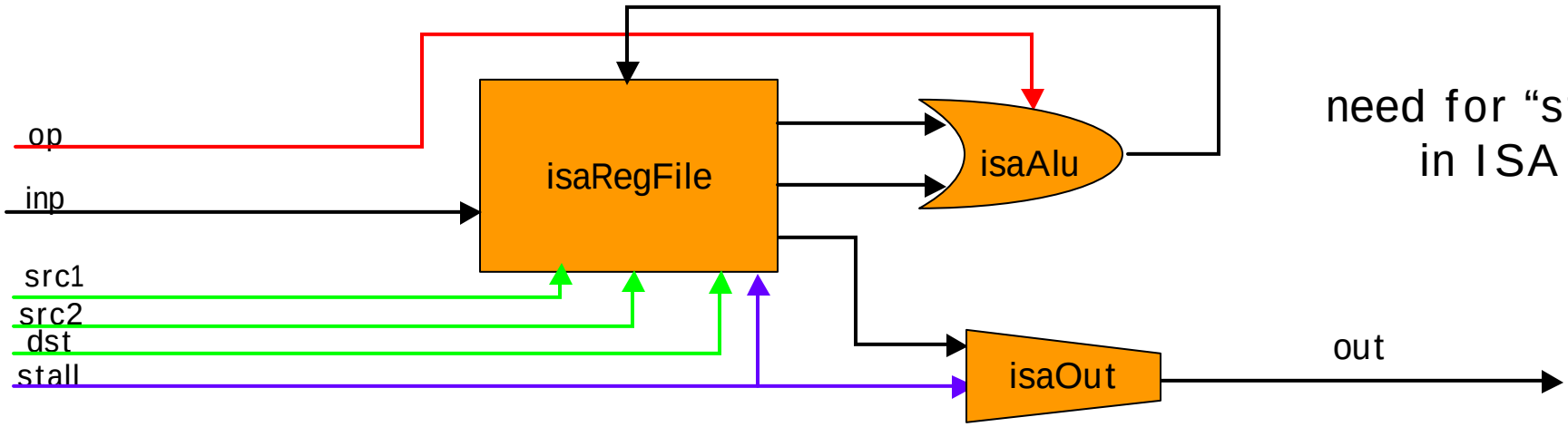

regFile
out
out
src1
src2
inp
op
dst
alu
P1
P2
FETCH
EXECUTE
WRITE-BACK


regFile
out
out
opr1
alu
opr2
res
src1
src2
inp
op
dst
p1inp
p1op
p1dst
p2op
p2dst
stall
stall

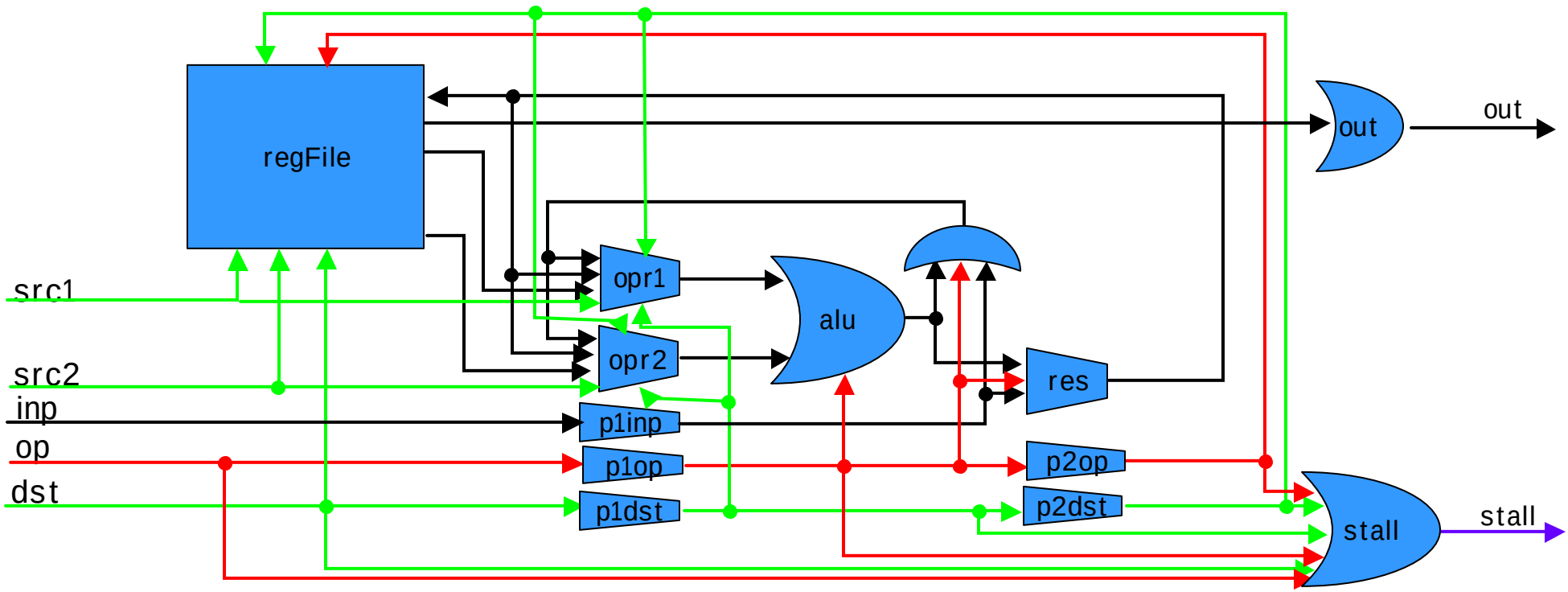


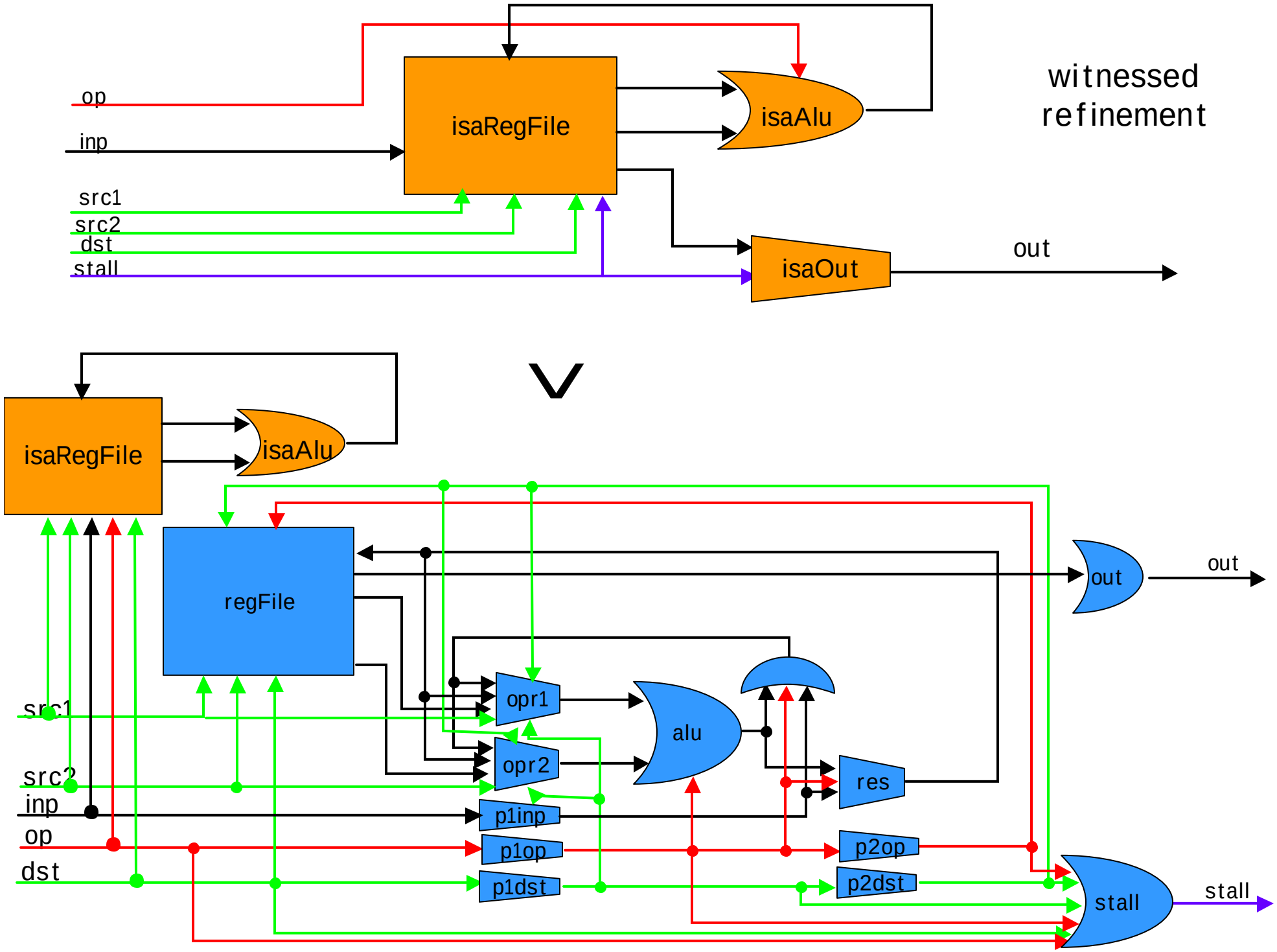
Goal: Establish
Pipeline refines ISA

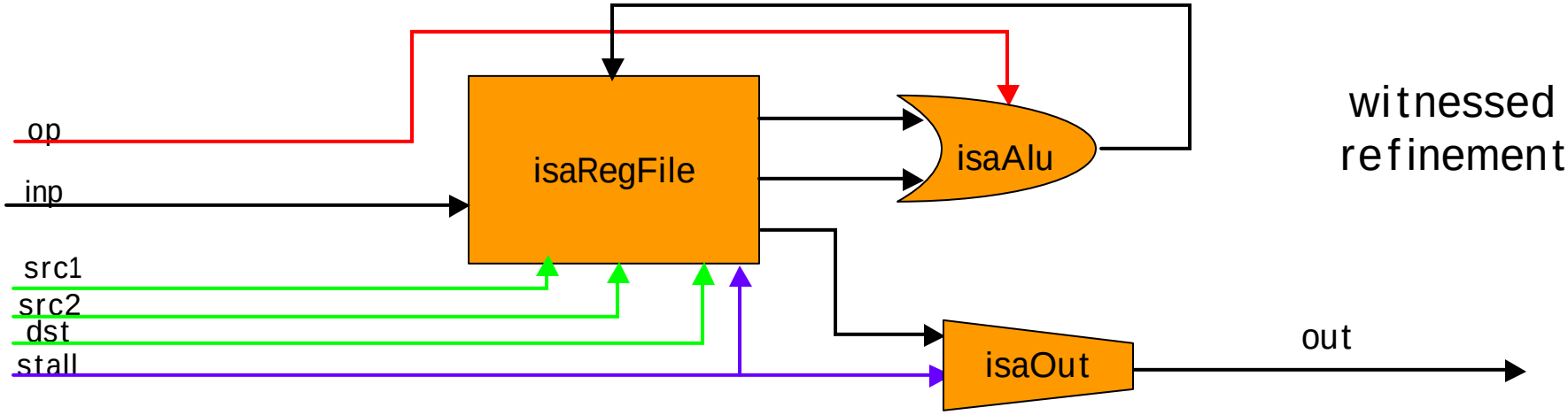




need for “stall” in ISA



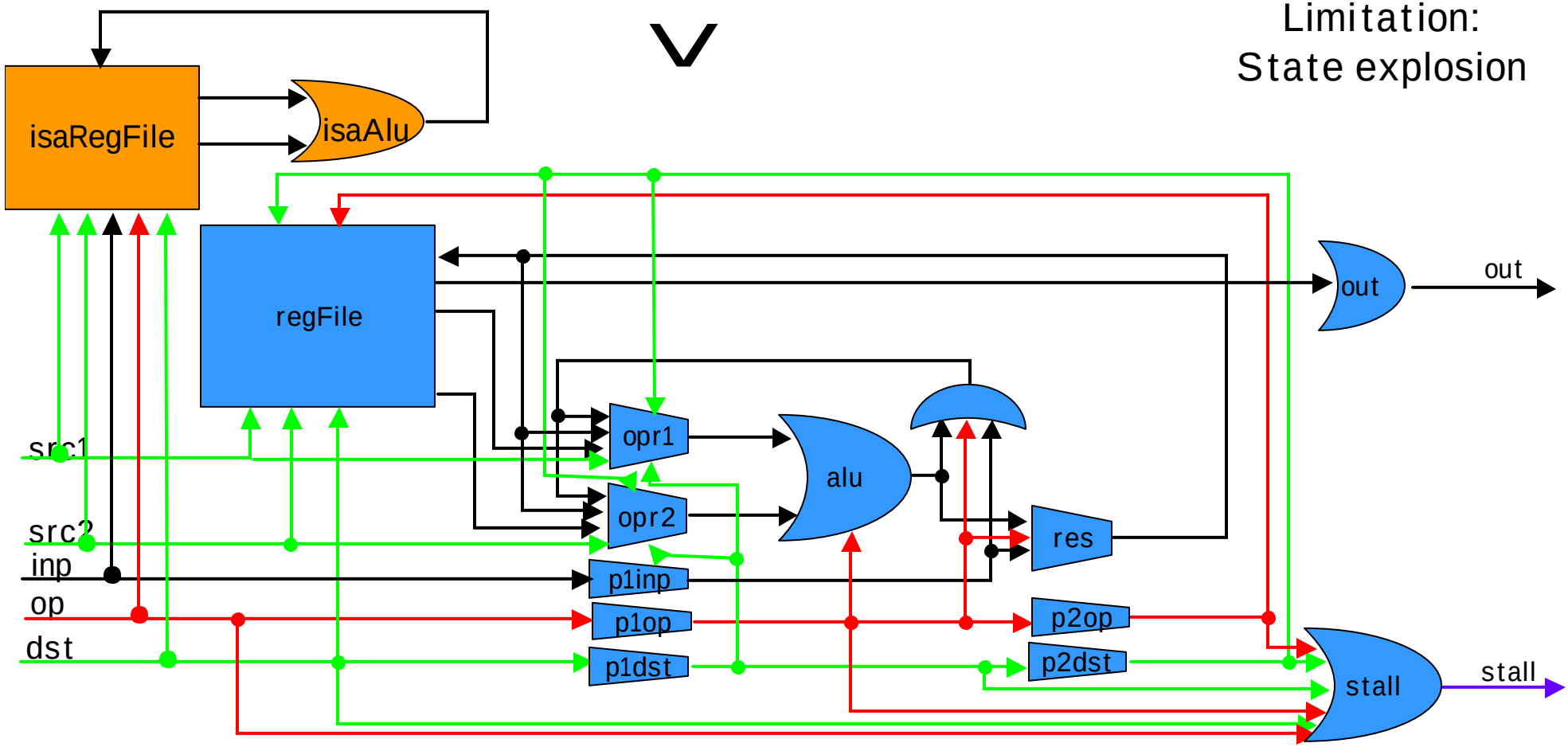

witnessed refinement
op
inp
src1
src2
dst
stall
isaRegFile
isaAlu
isaOut
out
V
isaRegFile
isaAlu
regFile
out
out
opr1
opr2
alu
res
p1inp
p1op
p1dst
p2op
p2dst
stall
stall
src1
src2
inp
op
dst

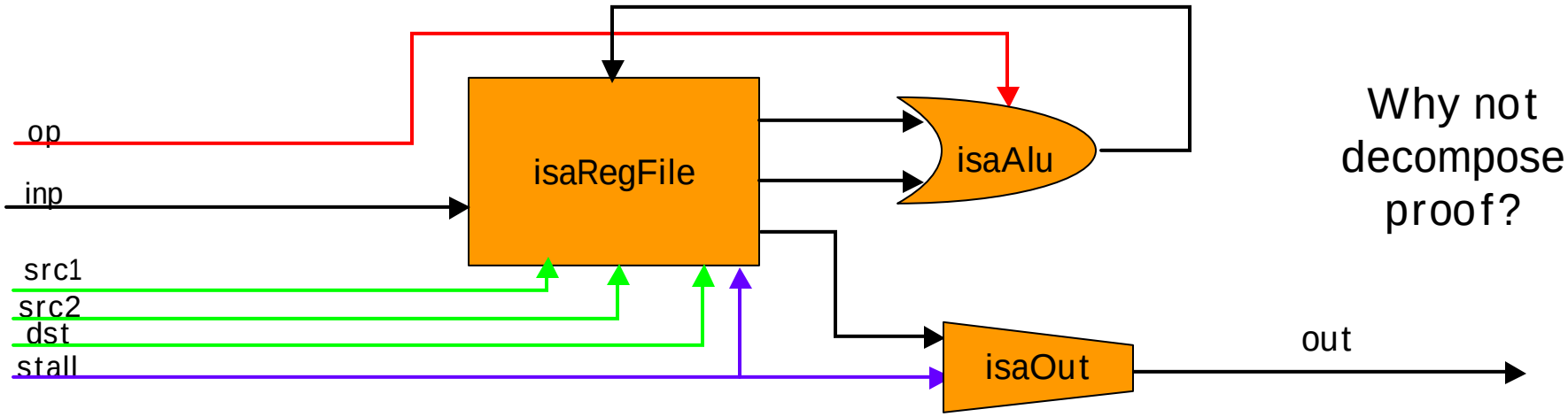


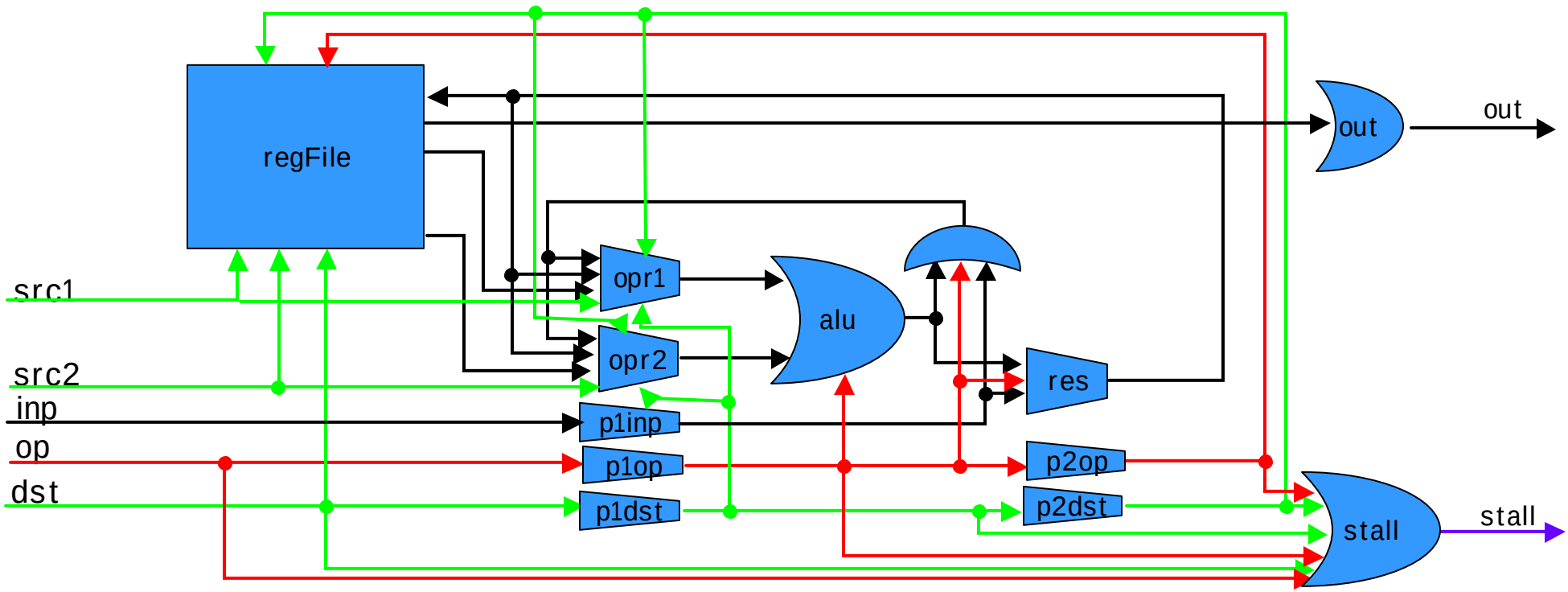
witnessed refinement

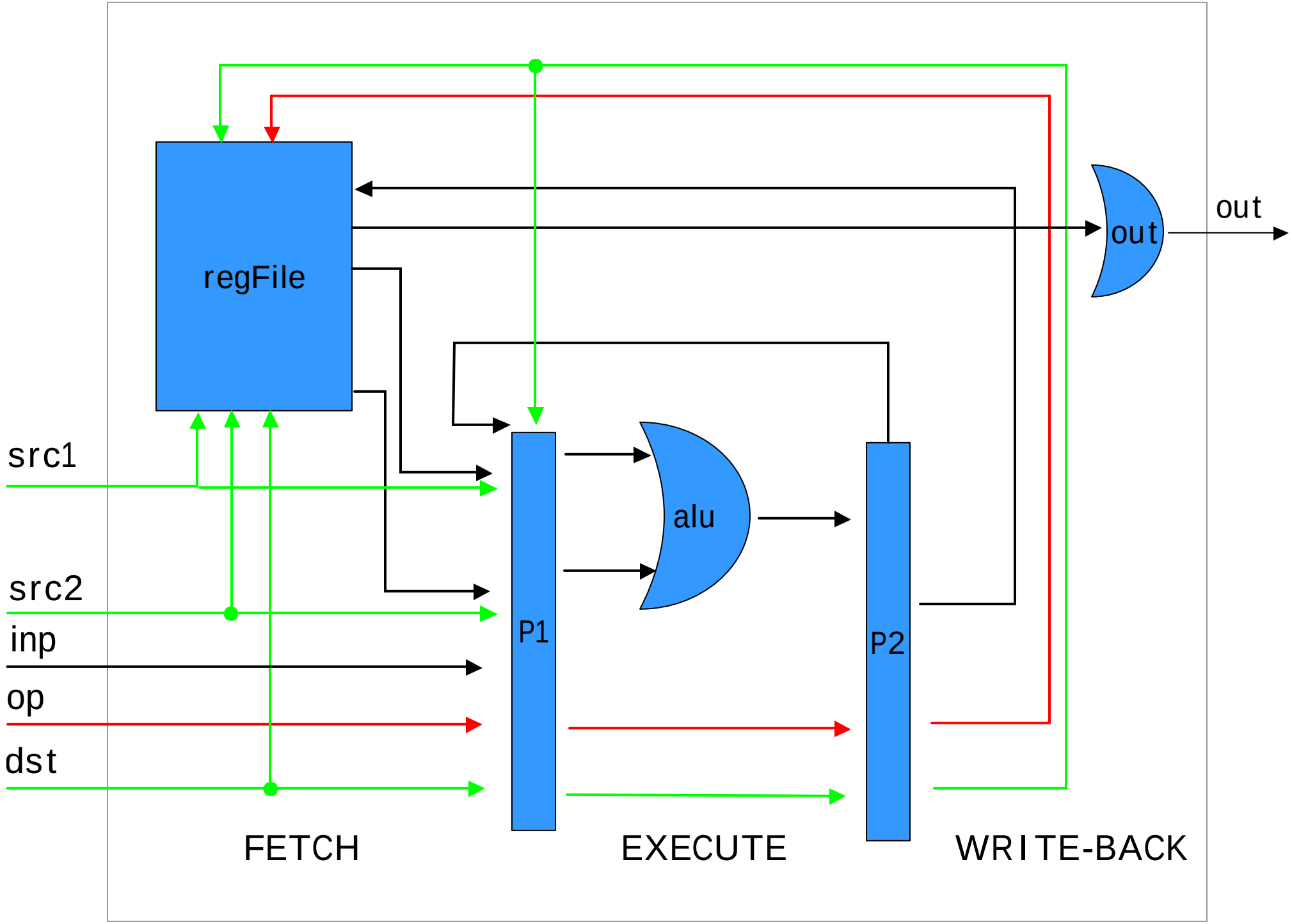
V

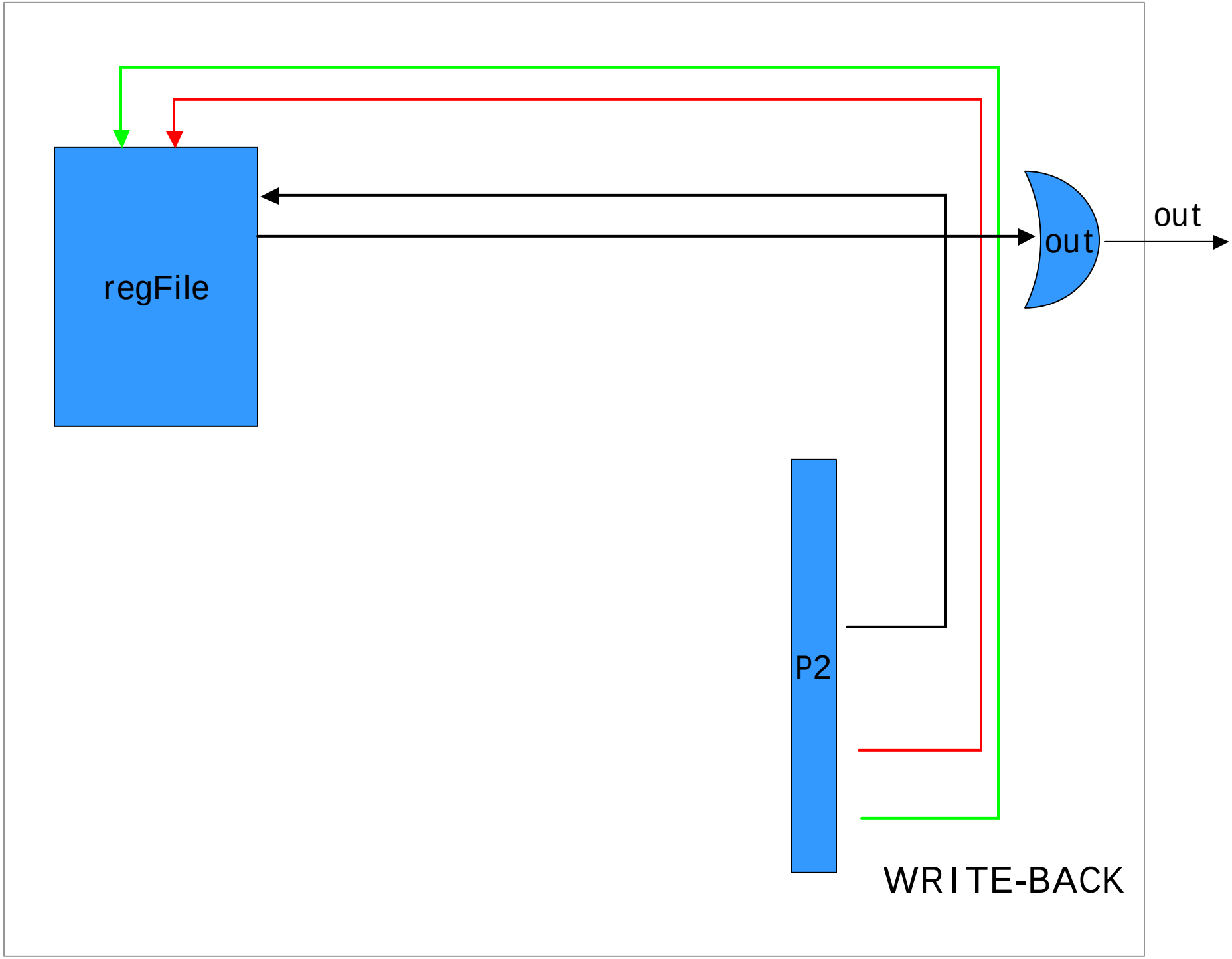
Limitation: State explosion

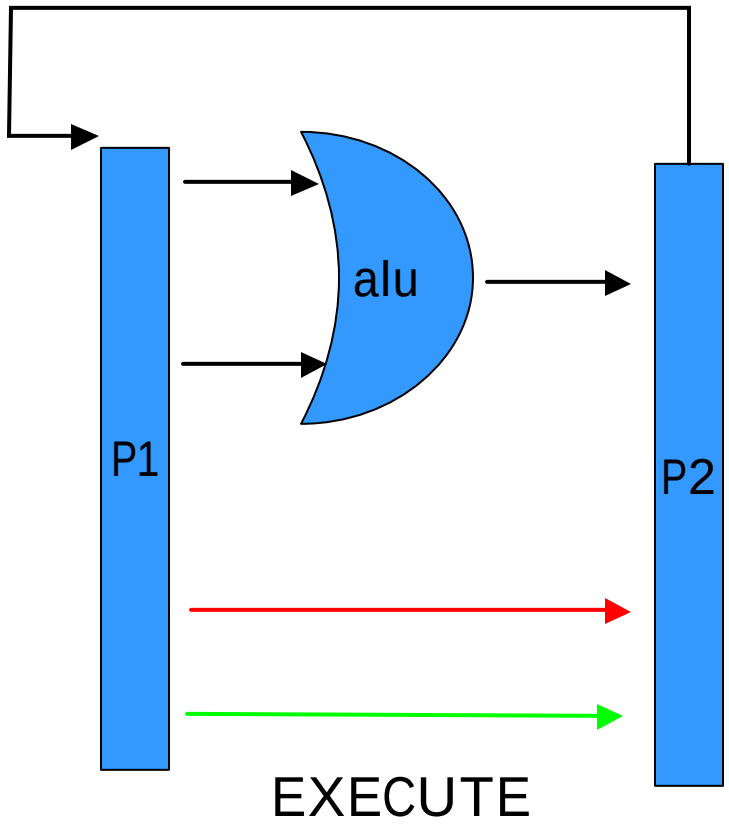


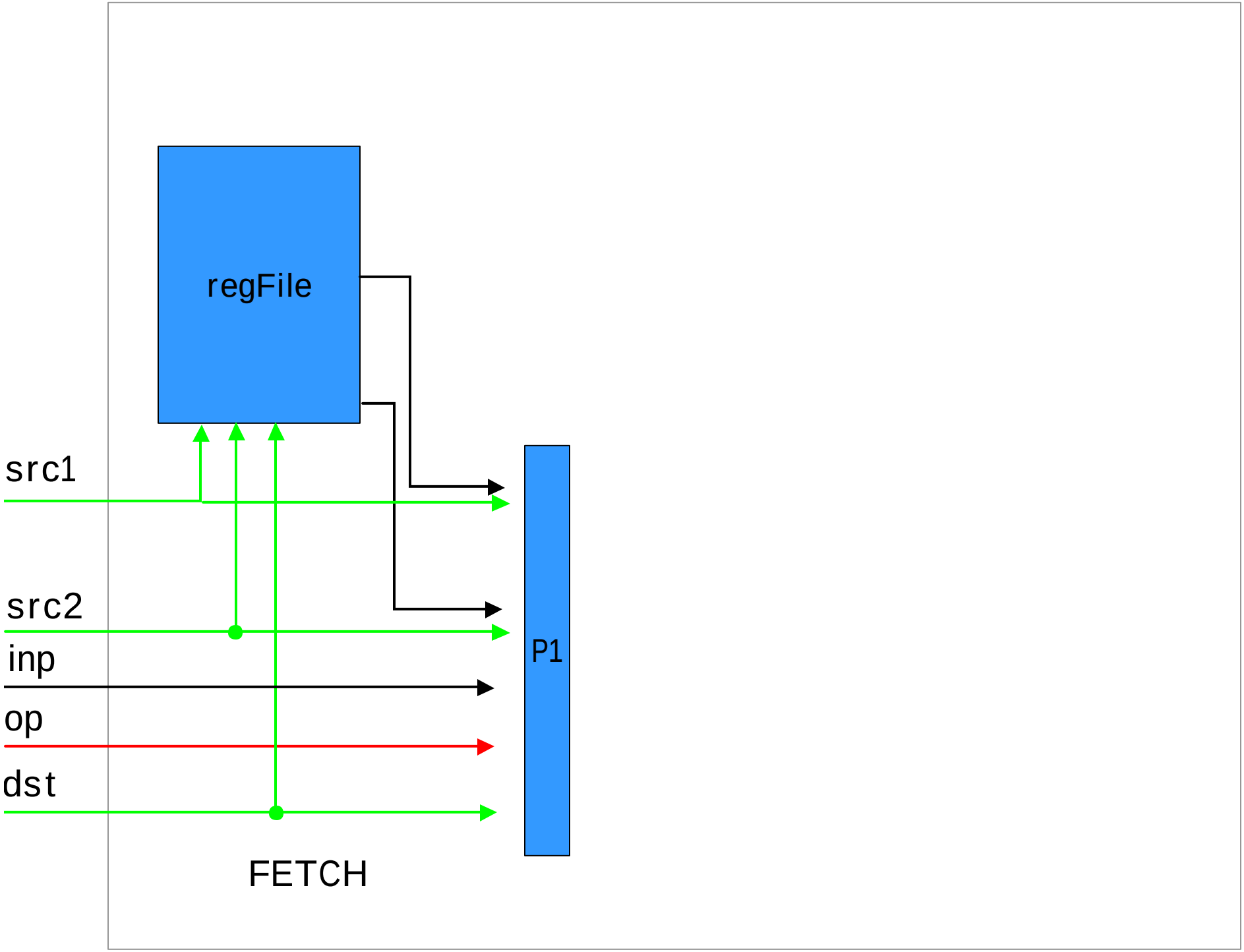

op
inp
src1
src2
dst
stall
isaRegFile
isaAlu
isaOut
out
Why not decompose proof?

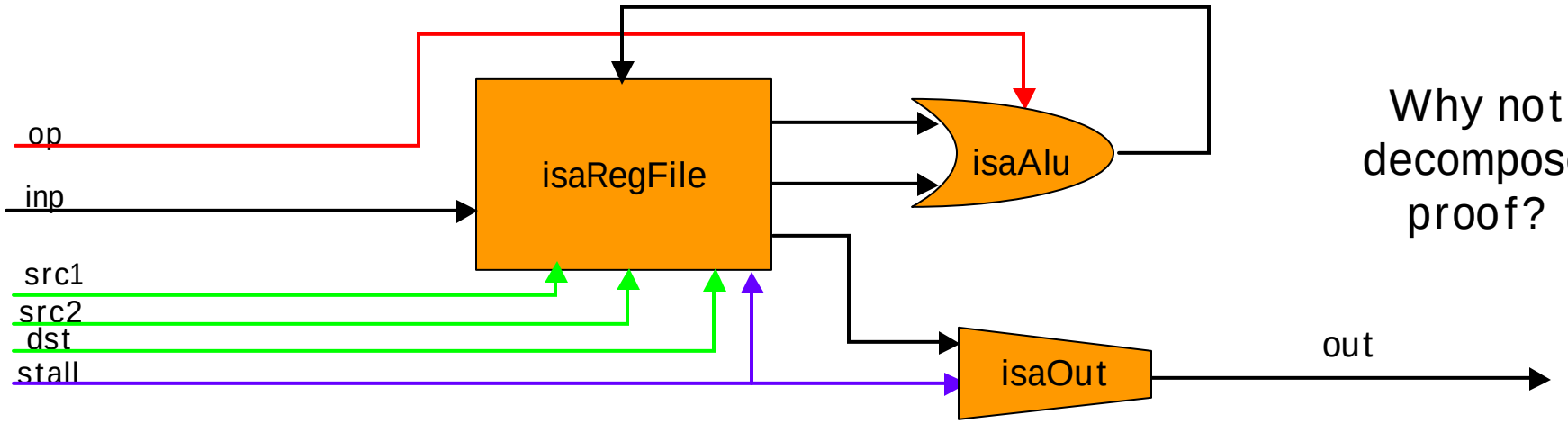

regFile
out
out
src1
src2
inp
op
dst
opr1
opr2
p1inp
p1op
p1dst
alu
res
p2op
p2dst
stall
stall


regFile
out
out
src1
src2
inp
op
dst
alu
P1
P2
FETCH
EXECUTE
WRITE-BACK

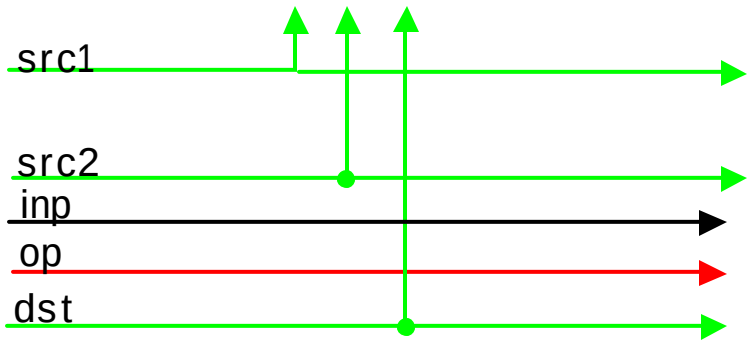

regFile
out
out
P2
WRITE-BACK

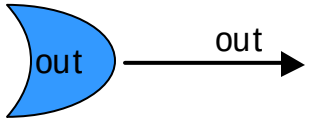

alu
P1
P2
EXECUTE

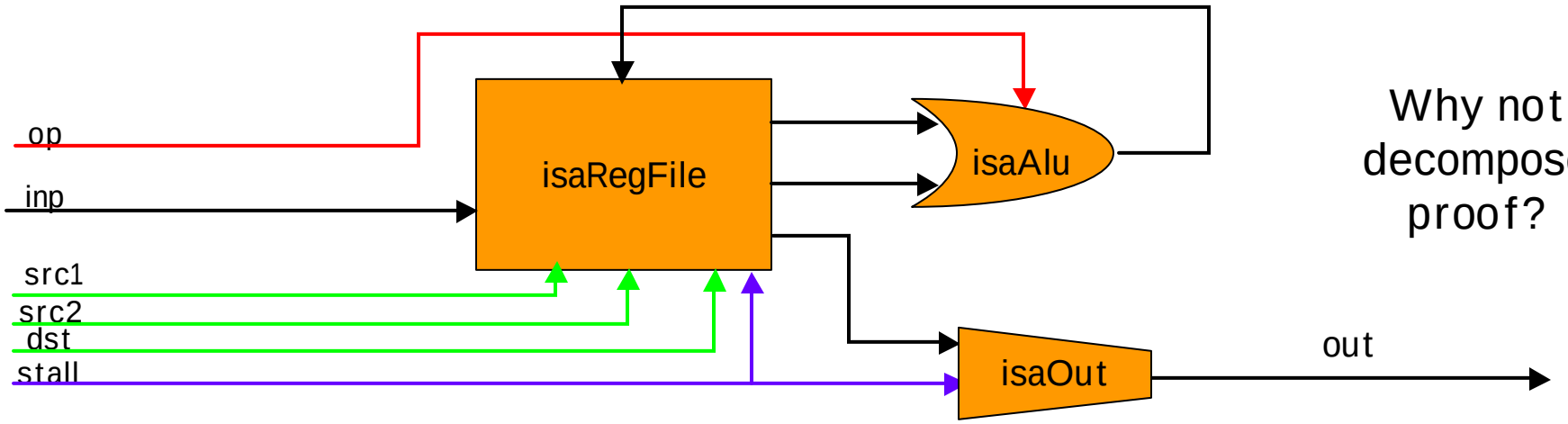

regFile
src1
src2
inp
op
dst
P1
FETCH



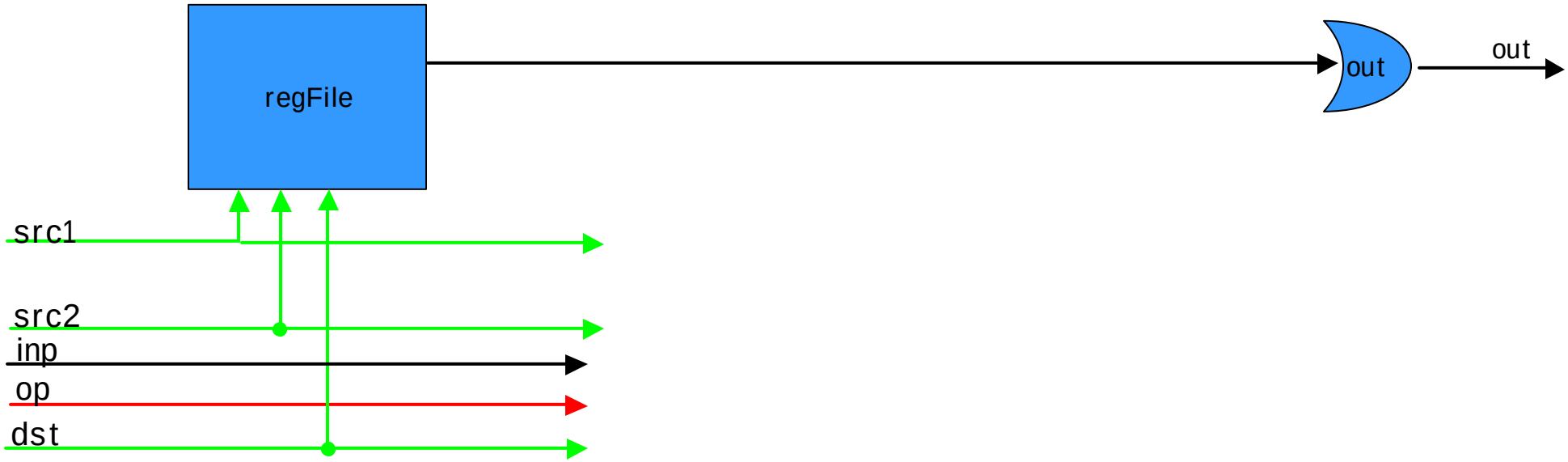
Why not decompose proof?

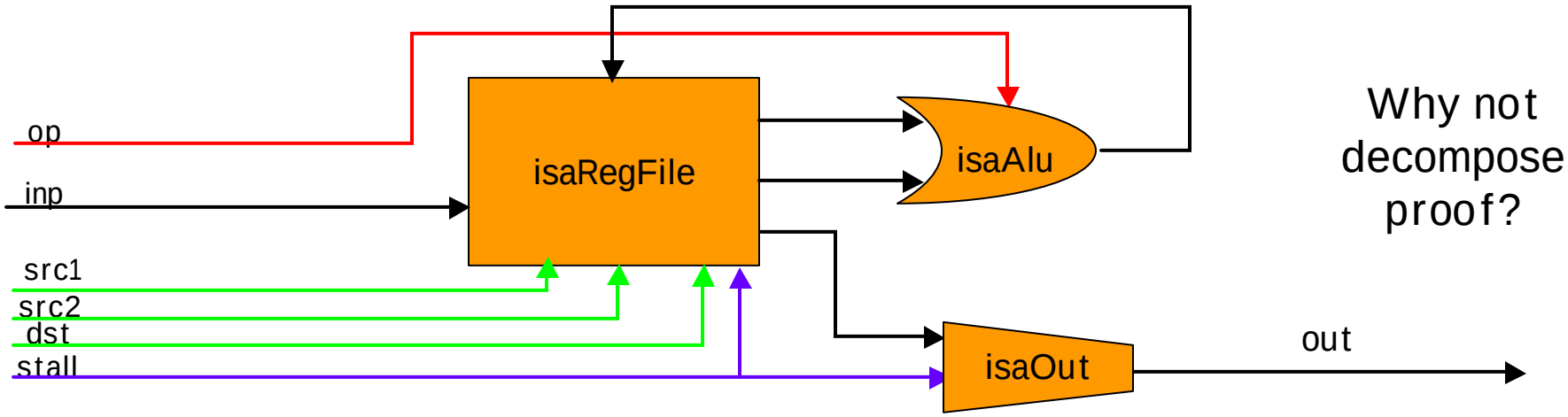




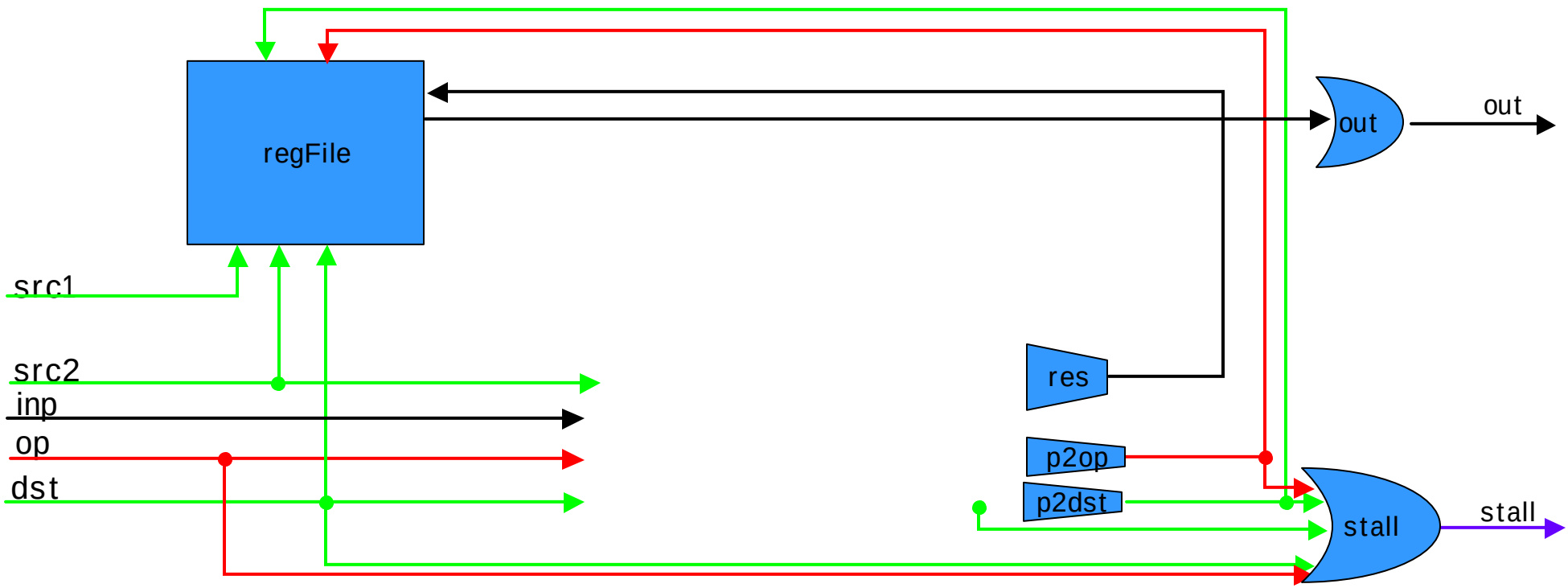


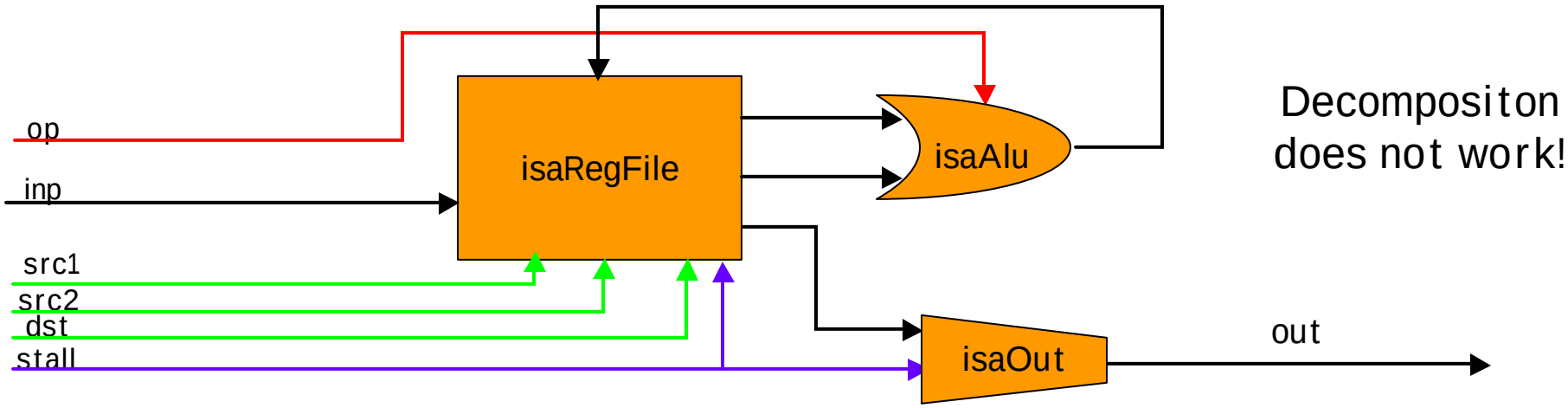
Why not decompose proof?

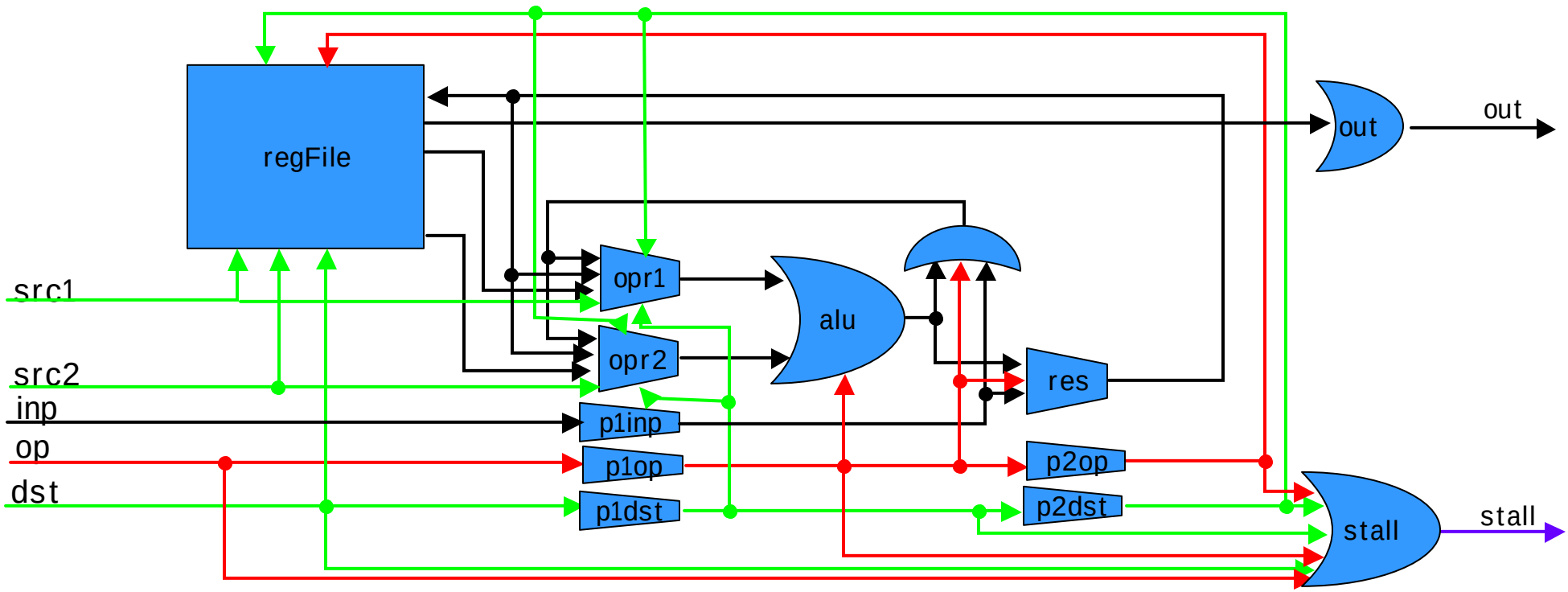


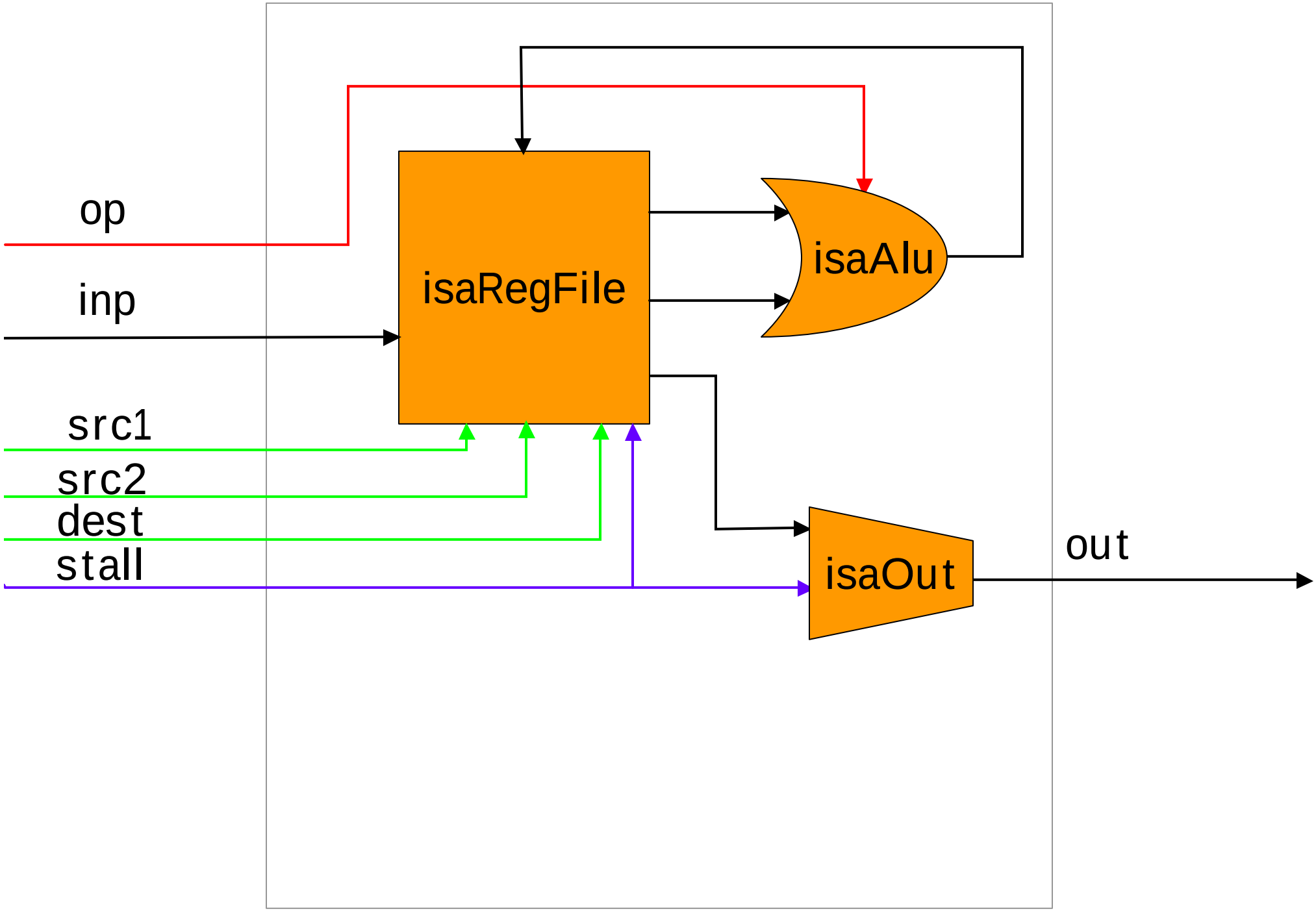


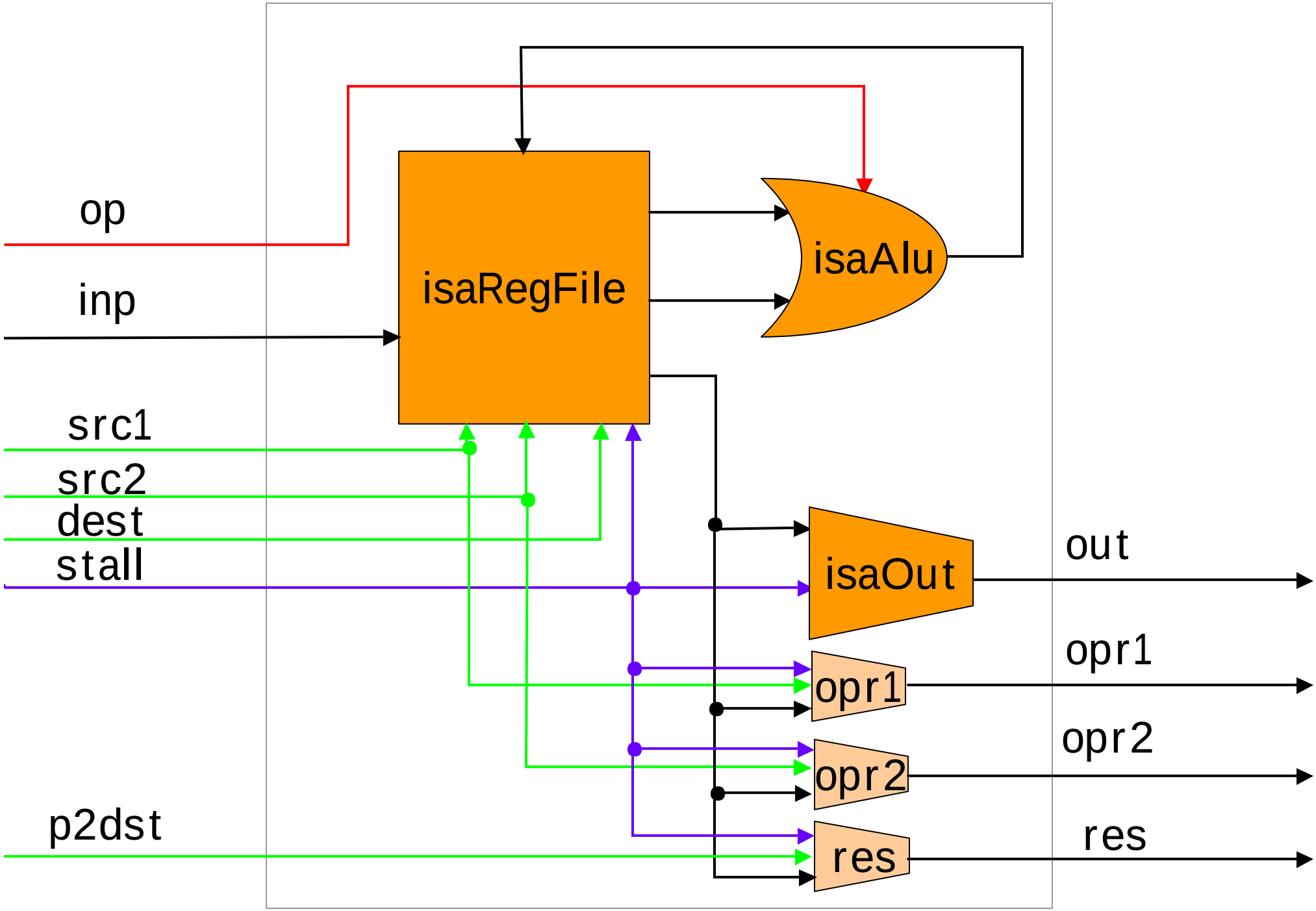
Why not decompose proof?

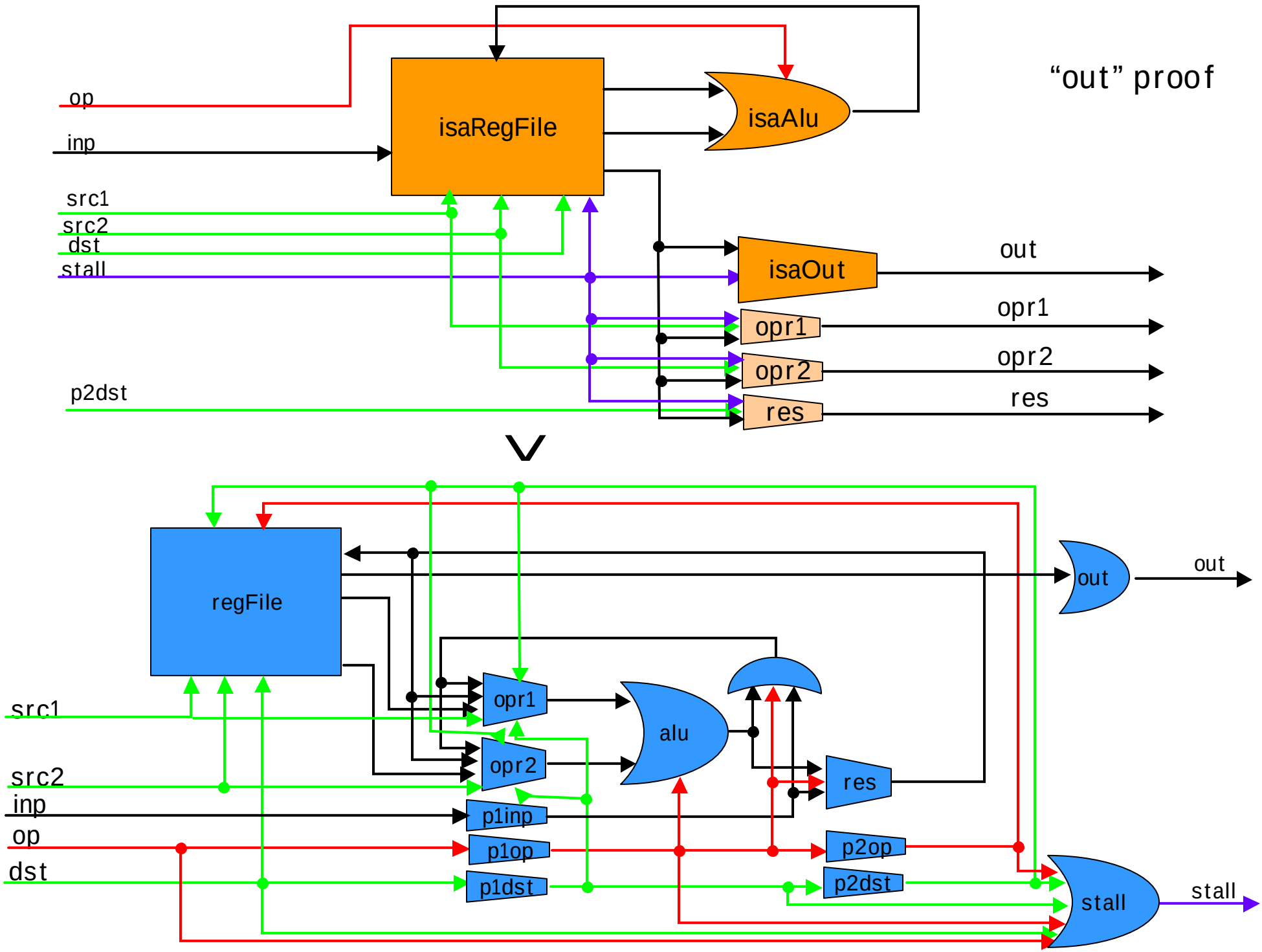



op
inp
src1
src2
dst
stall
isaRegFile
isaAlu
isaOut
out
Decompositon
does not work!

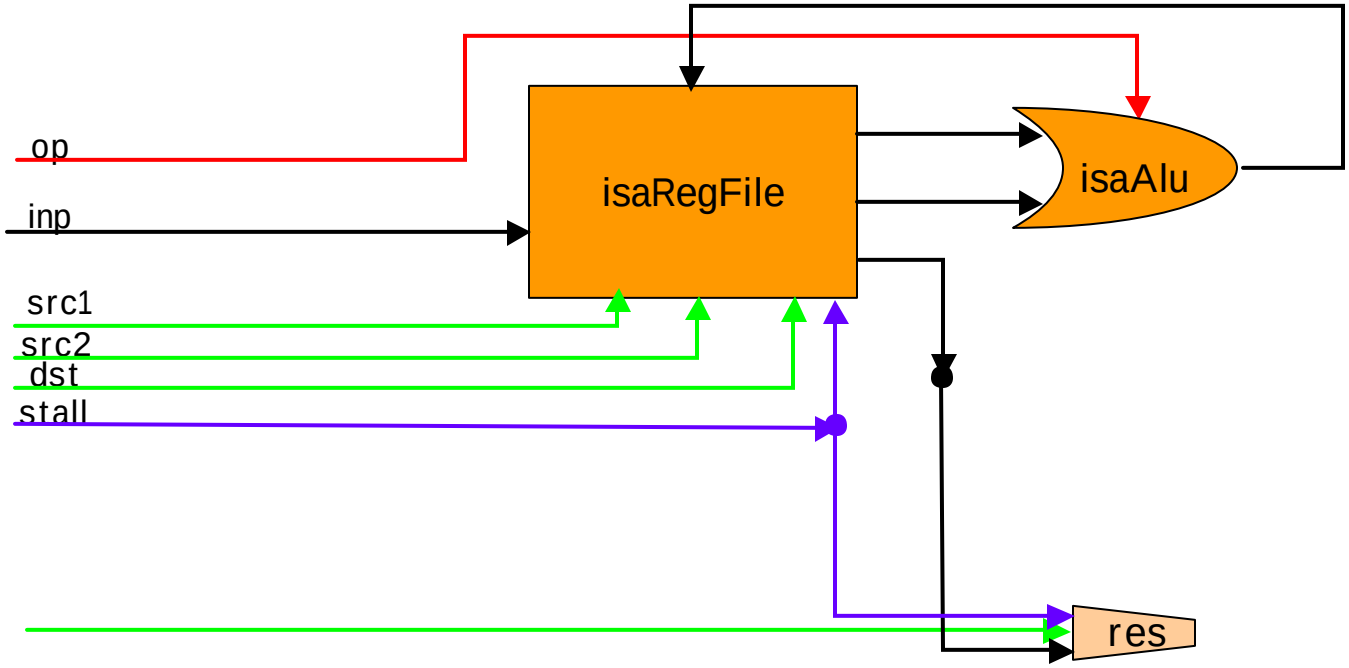

regFile
out
out
src1
src2
inp
op
dst
opr1
opr2
p1inp
p1op
p1dst
alu
res
p2op
p2dst
stall
stall

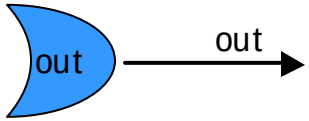

op
inp
src1
src2
dest
stall
isaRegFile
isaAlu
isaOut
out

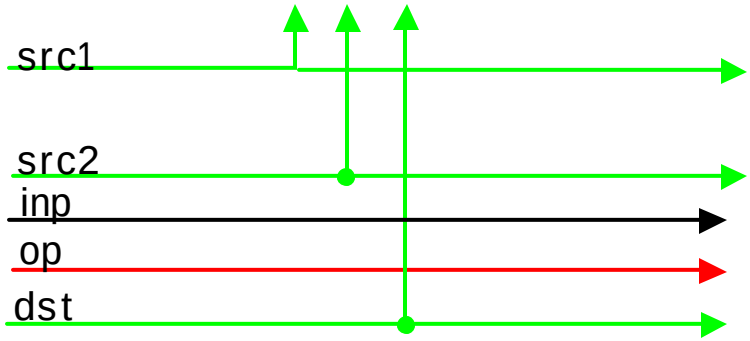

op
inp
src1
src2
dest
stall
p2dst
isaRegFile
isaAlu
isaOut
opr1
opr2
res
out
opr1
opr2
res

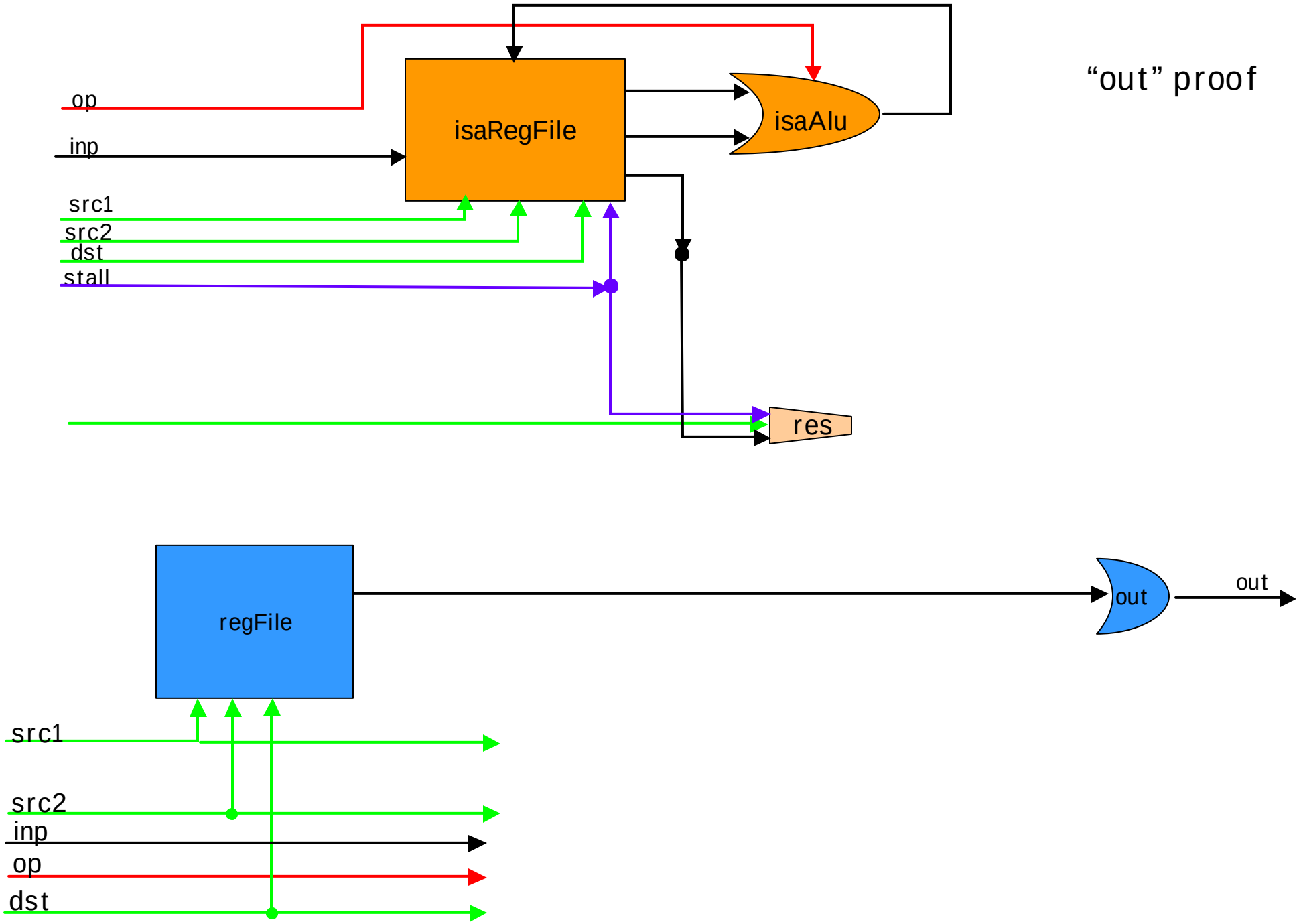

"out" proof
op
inp
isaRegFile
isaAlu
src1
src2
dst
stall
isaOut
out
opr1
opr1
opr2
opr2
p2dst
res
res
regFile
out
out
opr1
alu
src1
opr2
res
src2
inp
p1inp
op
p1op
p2op
dst
p1dst
p2dst
stall
stall

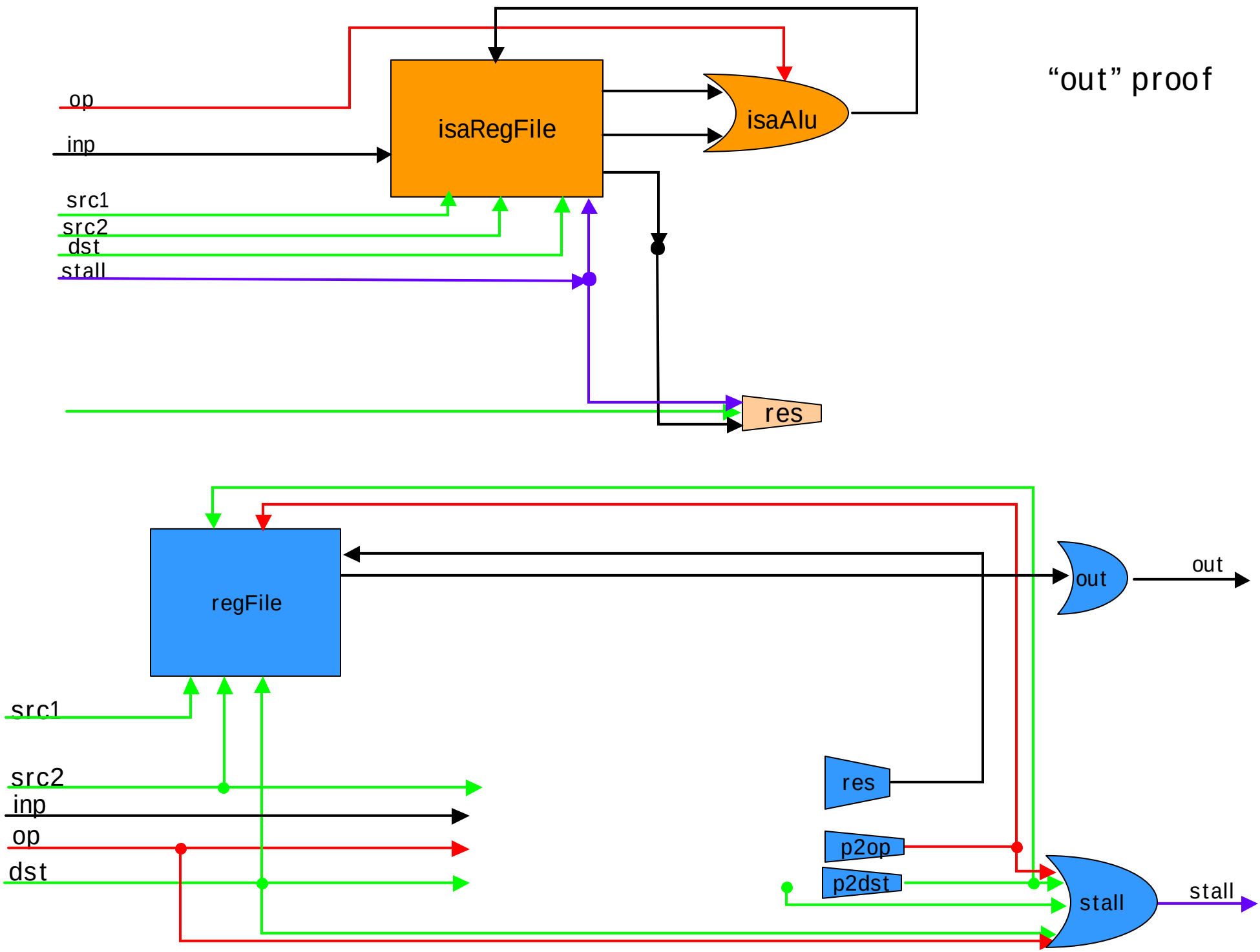
“out” proof

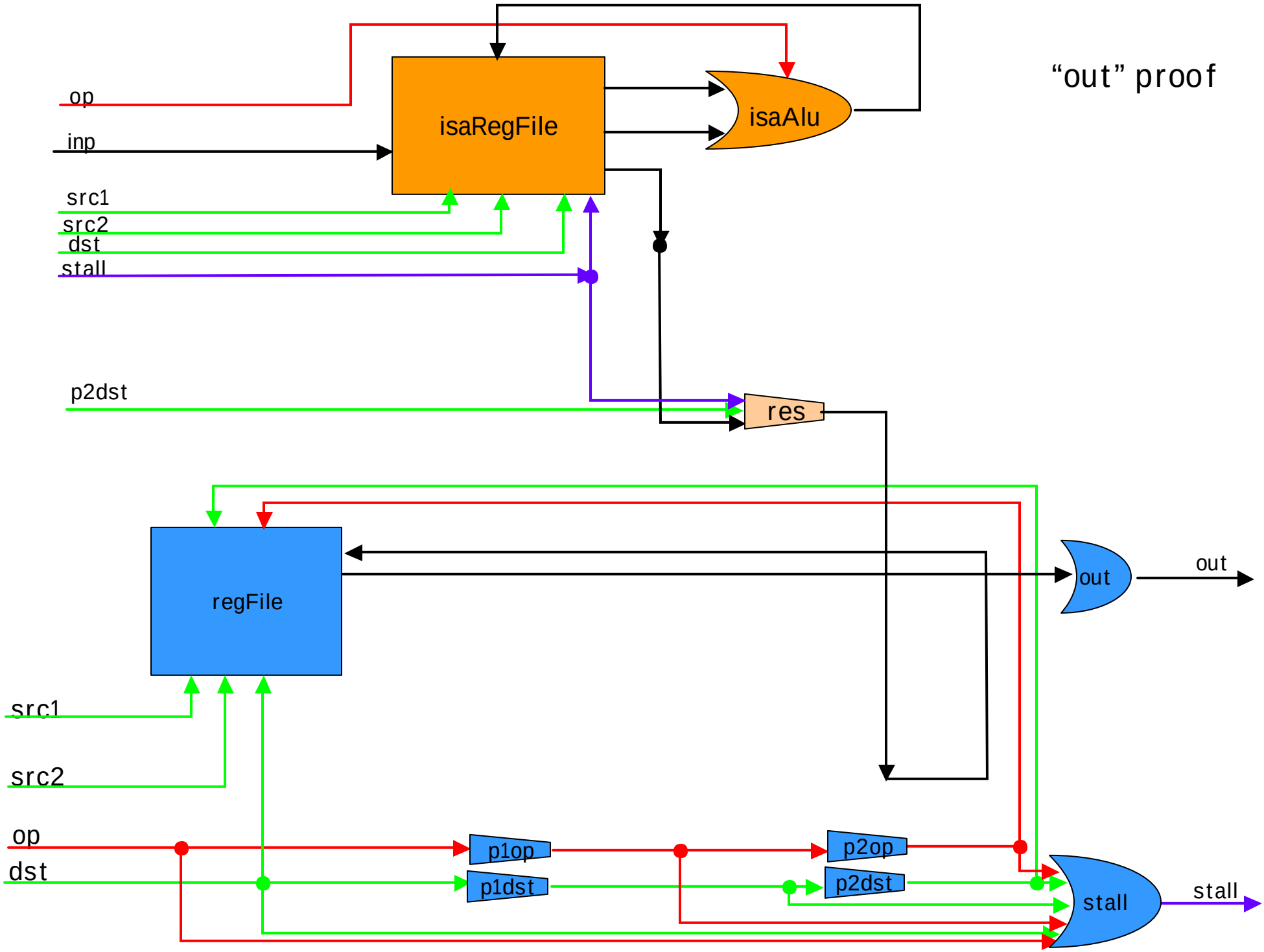

op
inp
src1
src2
dst
stall
isaRegFile
isaAlu
res

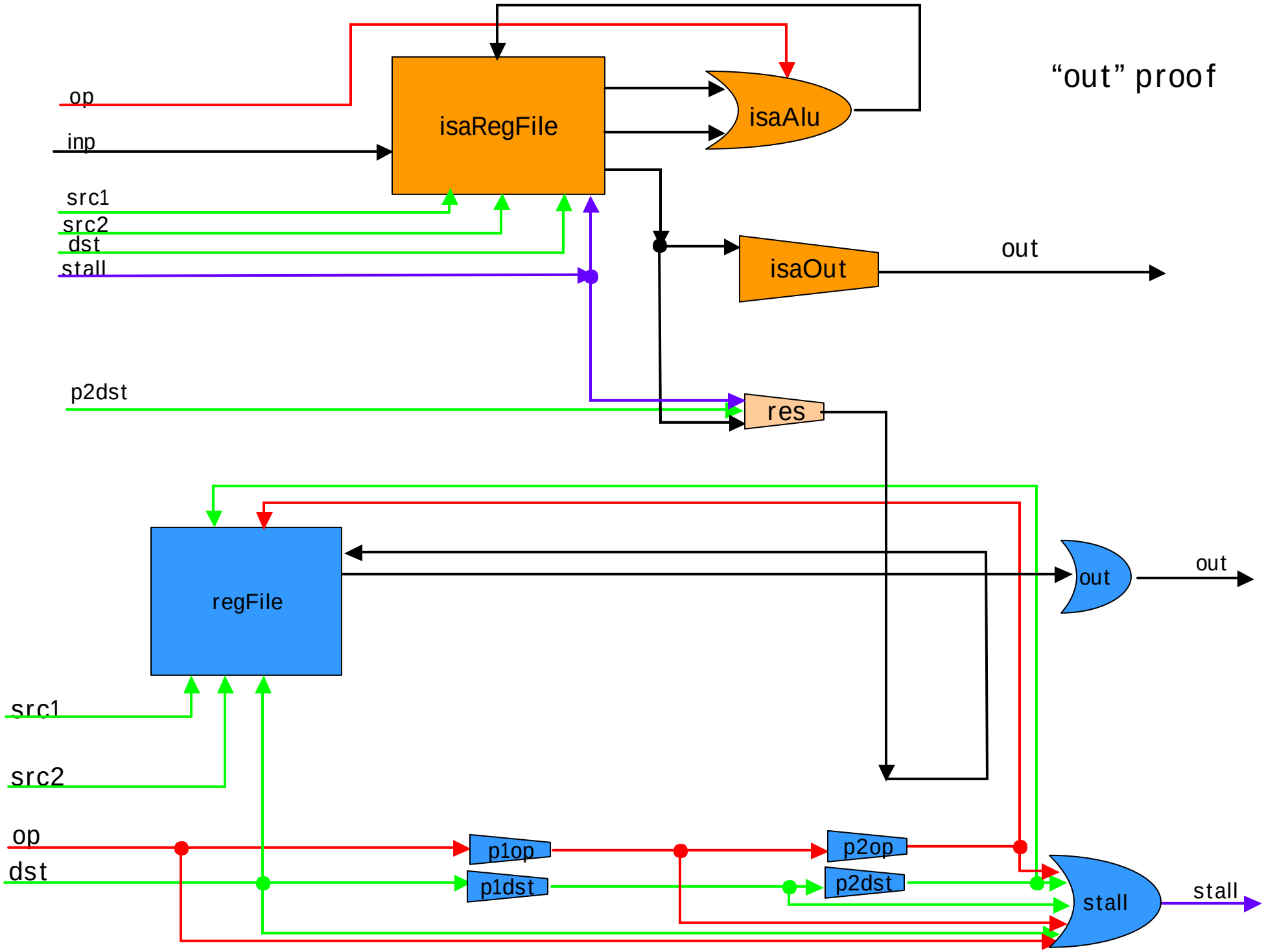

out
out

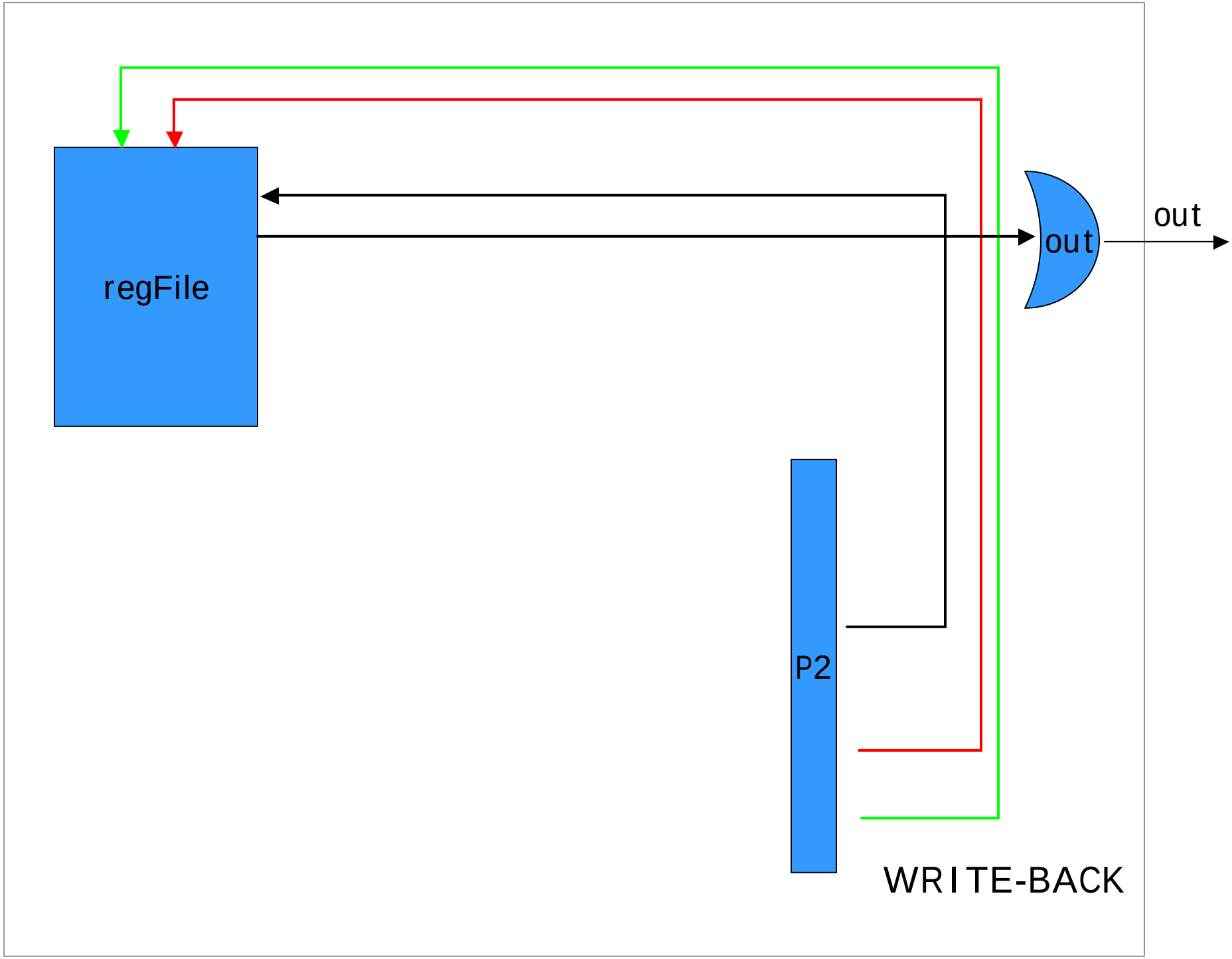

src1
src2
inp
op
dst

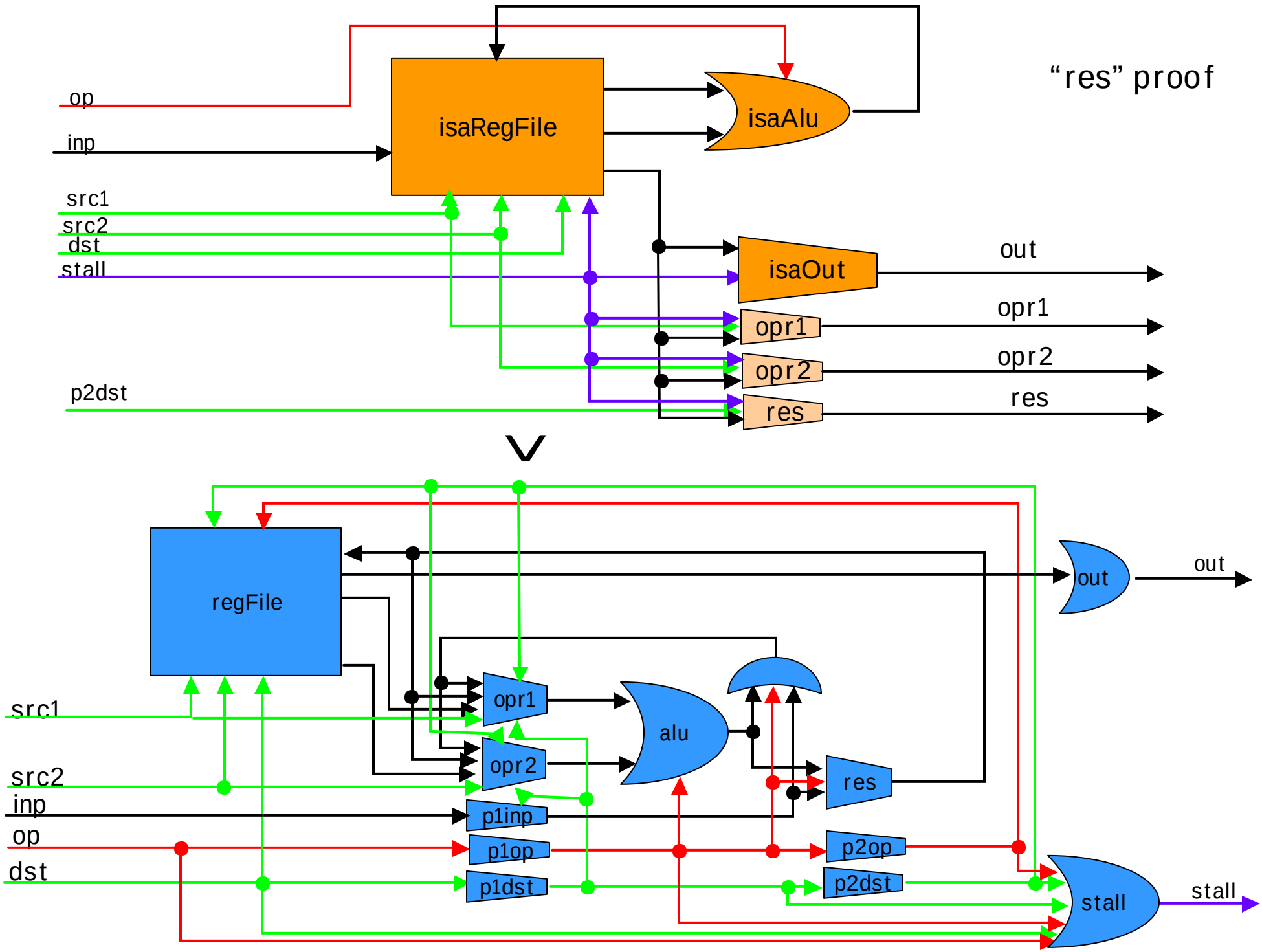

"out" proof
op
inp
src1
src2
dst
stall
isaRegFile
isaAlu
res
regFile
out
out
src1
src2
inp
op
dst

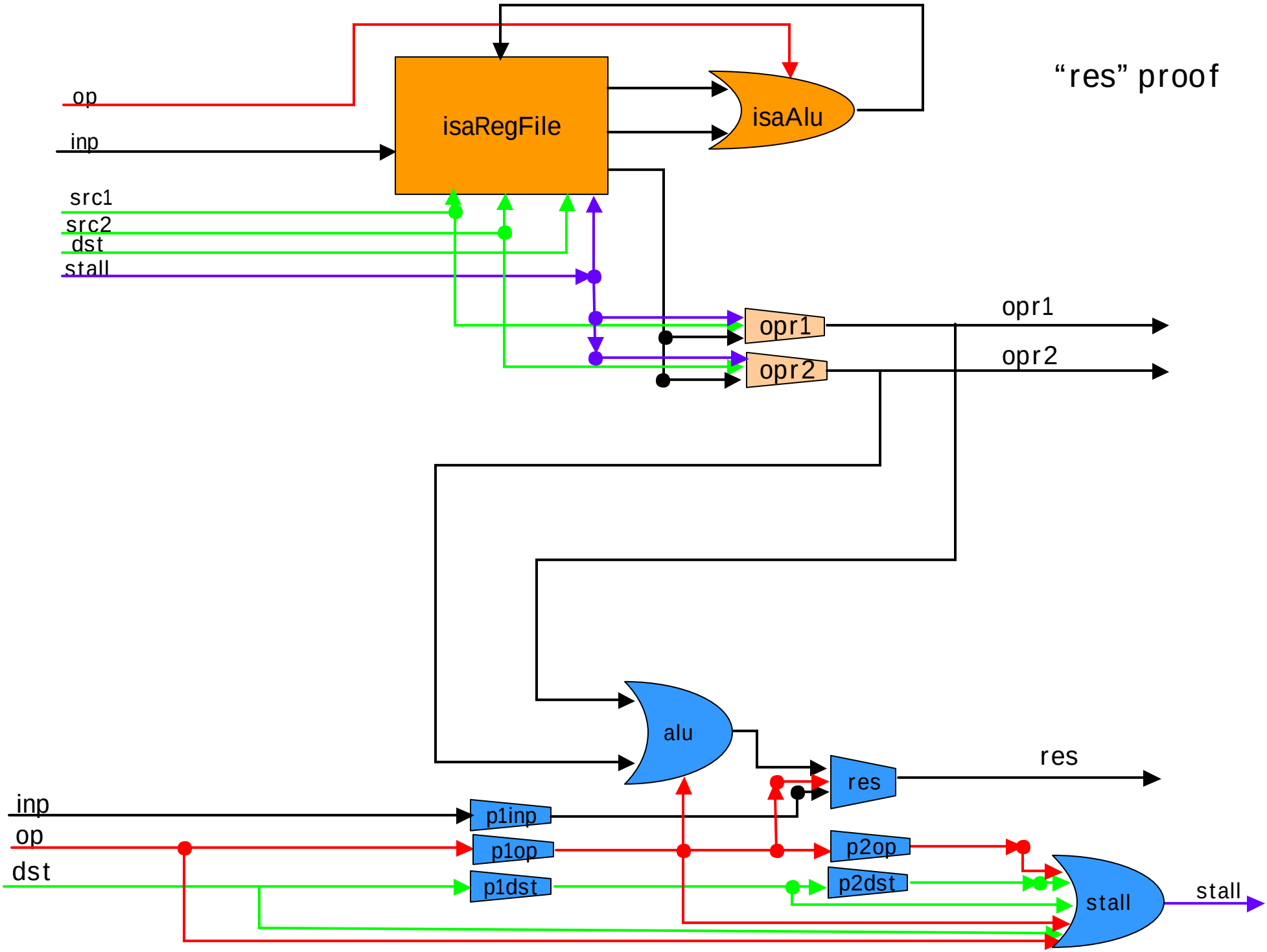

"out" proof
op
inp
src1
src2
dst
stall
isaRegFile
isaAlu
res
regFile
out
out
src1
src2
inp
op
dst
res
p2op
p2dst
stall
stall

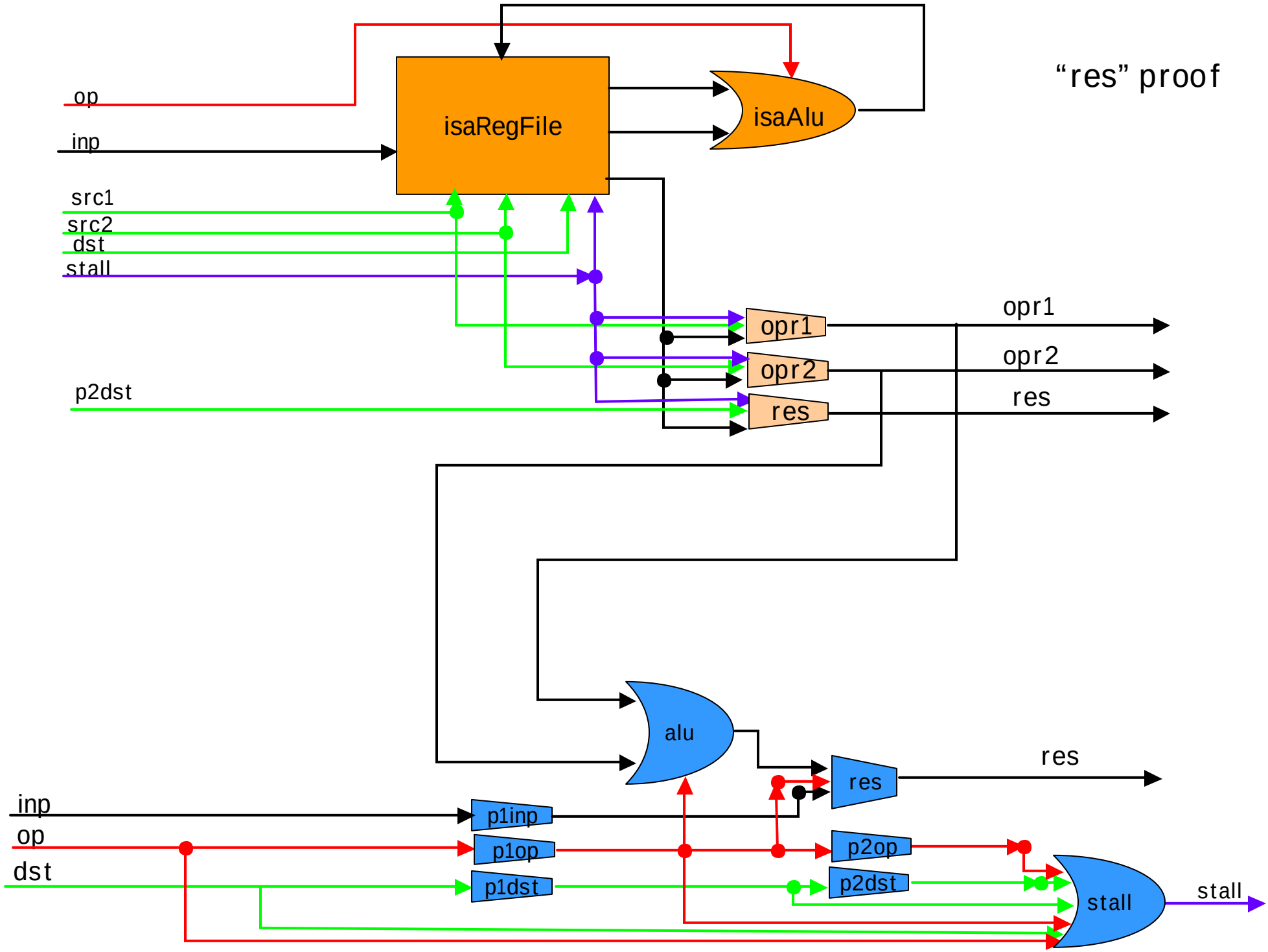

"out" proof
op
inp
isaRegFile
isaAlu
src1
src2
dst
stall
p2dst
res
regFile
out
out
src1
src2
op
dst
p1op
p1dst
p2op
p2dst
stall
stall

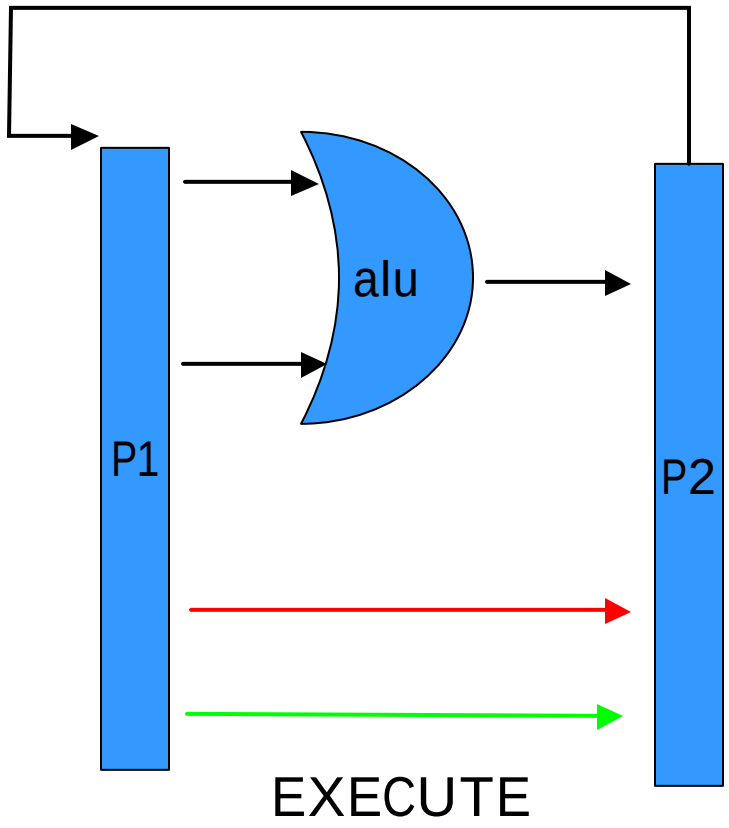

"out" proof
op
inp
isaRegFile
isaAlu
src1
src2
dst
stall
isaOut
out
p2dst
res
regFile
out
out
src1
src2
op
dst
p1op
p1dst
p2op
p2dst
stall
stall

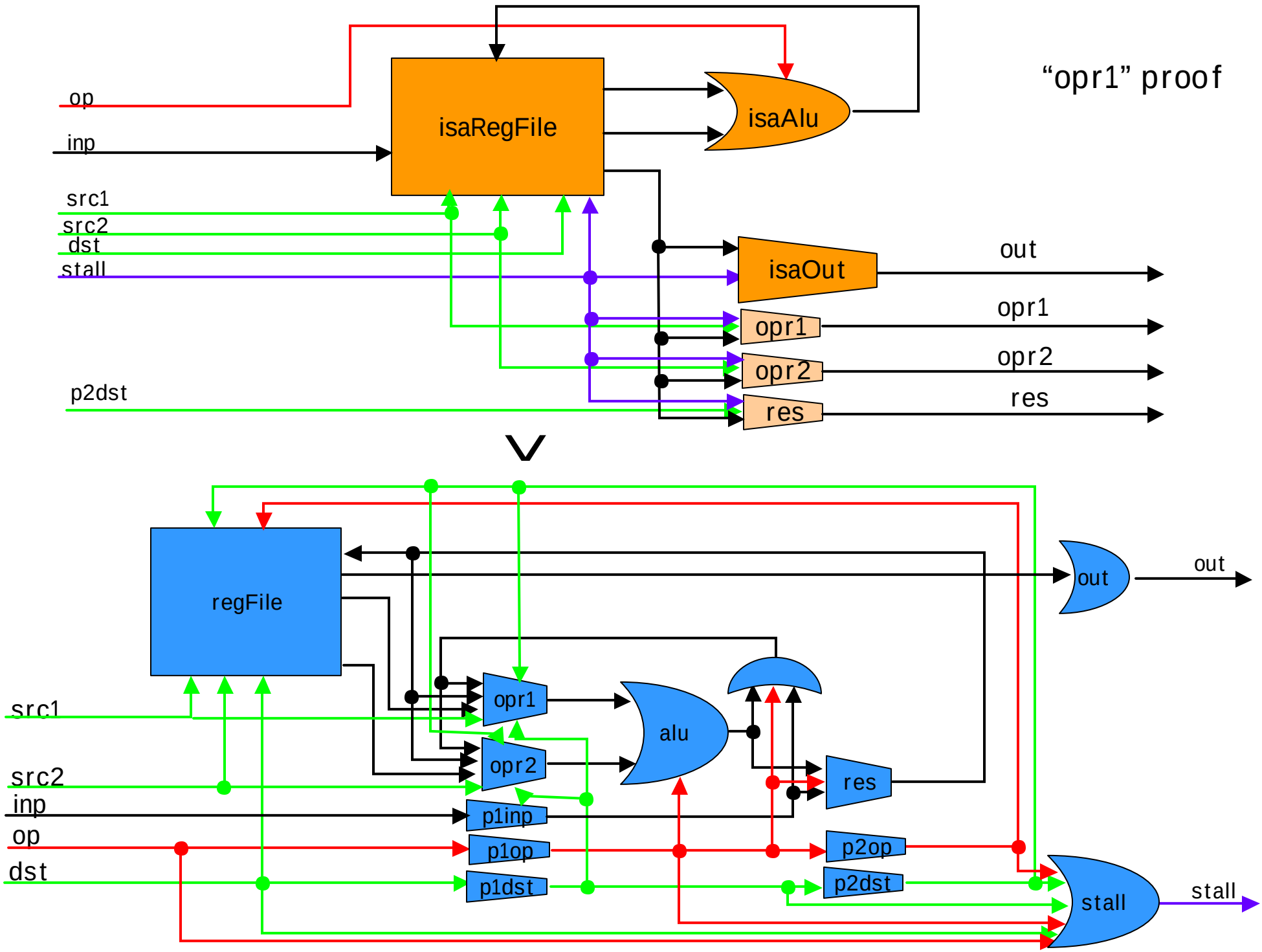

regFile
out
out
P2
WRITE-BACK

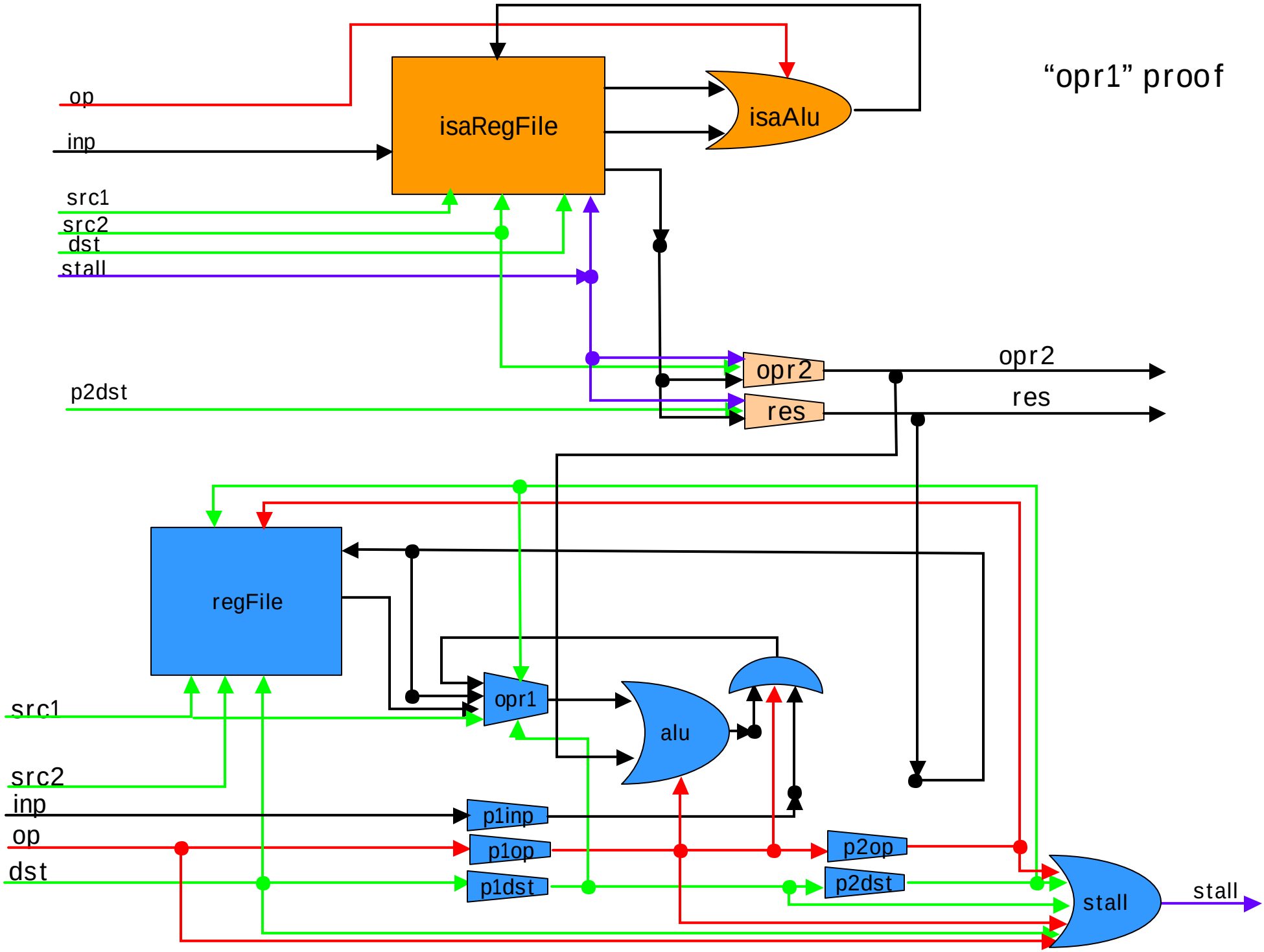

"res" proof
op
inp
isaRegFile
isaAlu
src1
src2
dst
stall
isaOut
out
opr1
opr1
opr2
opr2
p2dst
res
res
regFile
out
out
opr1
alu
src1
opr2
res
src2
inp
p1inp
op
p1op
p2op
dst
p1dst
p2dst
stall
stall

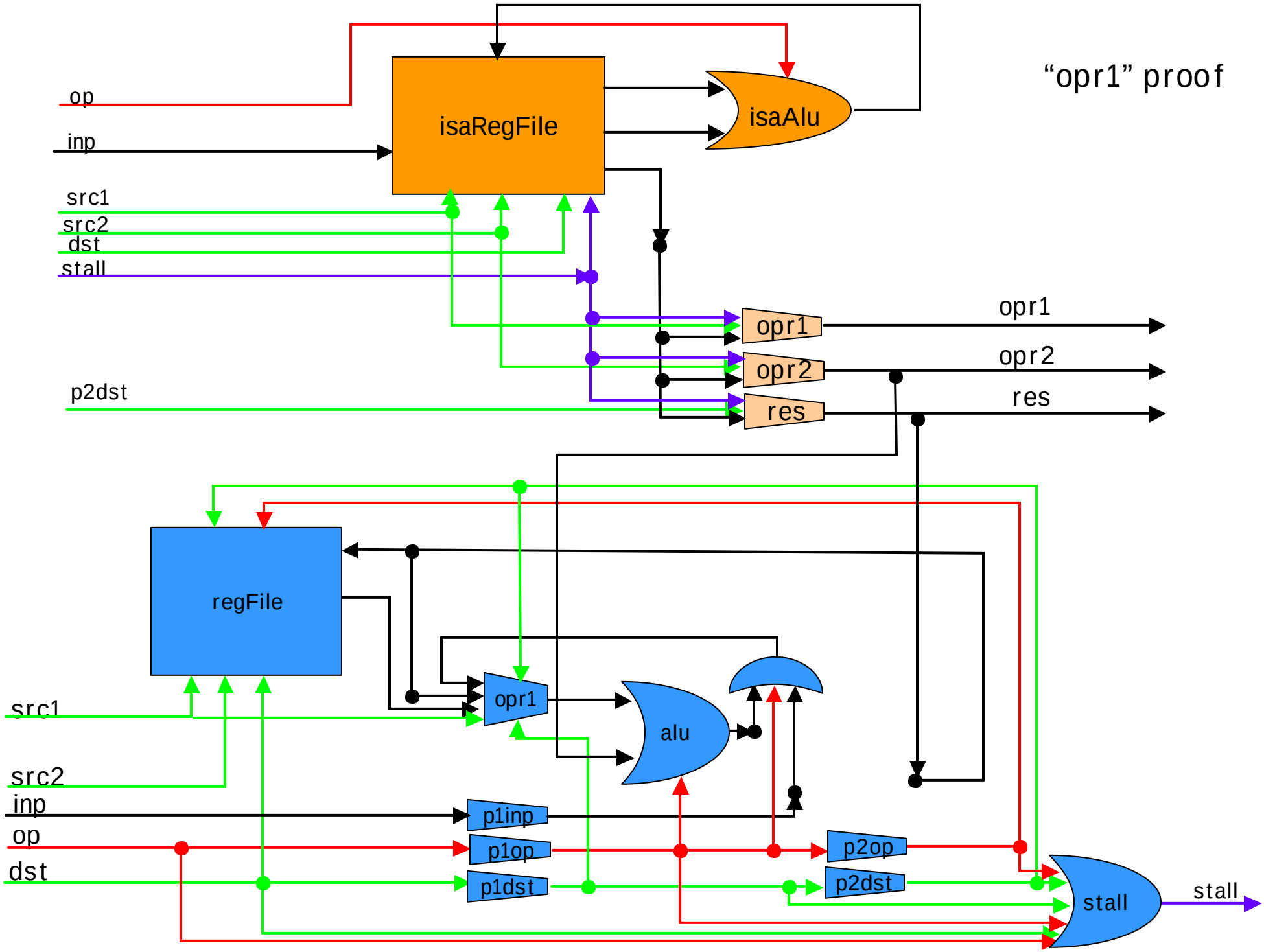

"res" proof
op
inp
isaRegFile
isaAlu
src1
src2
dst
stall
opr1
opr2
opr1
opr2
alu
res
res
inp
op
dst
p1inp
p1op
p1dst
p2op
p2dst
stall
stall

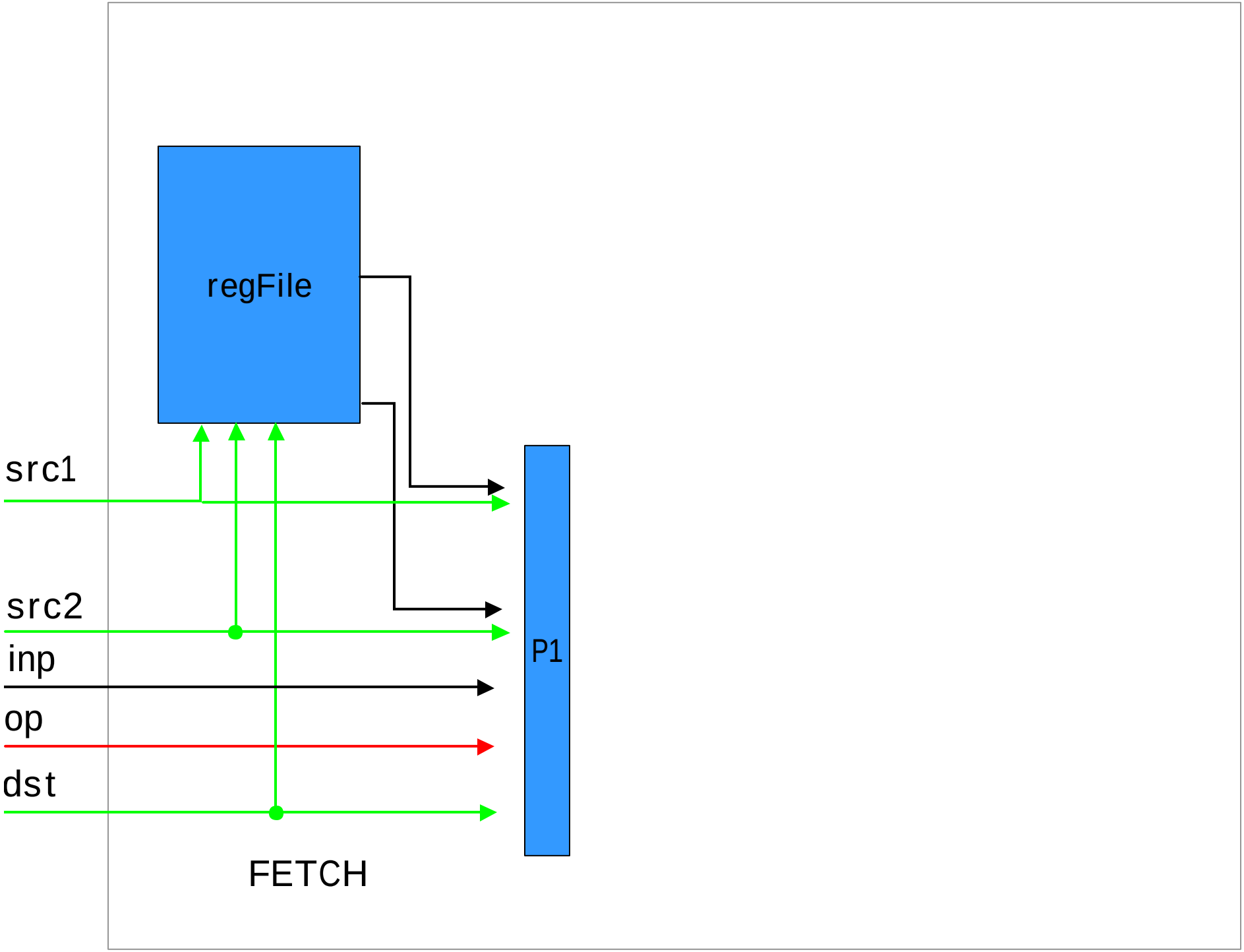

"res" proof
op
inp
isaRegFile
isaAlu
src1
src2
dst
stall
opr1
opr2
res
p2dst
opr1
opr2
res
alu
res
res
inp
op
dst
p1inp
p1op
p1dst
p2op
p2dst
stall
stall

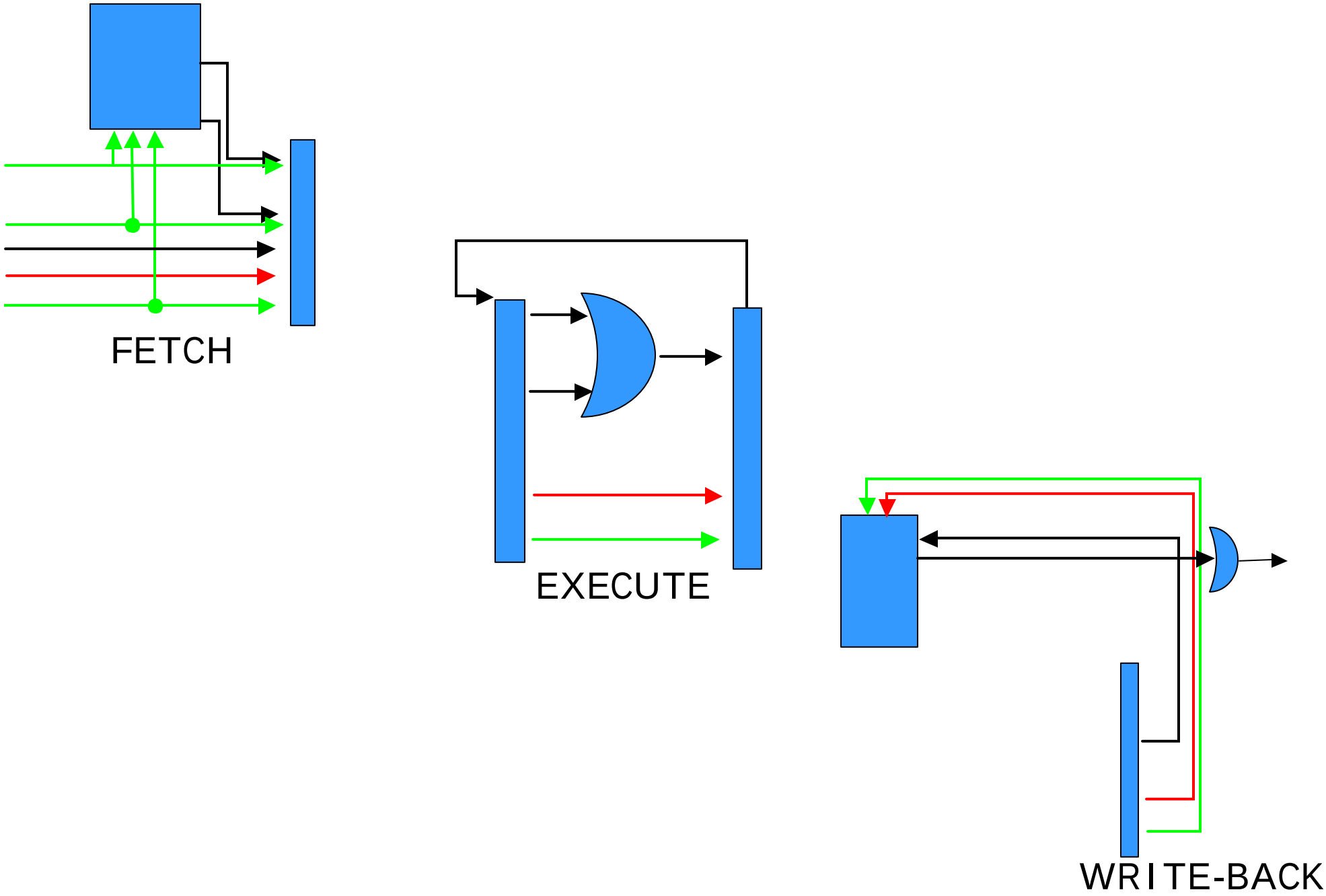

alu
P1
P2
EXECUTE

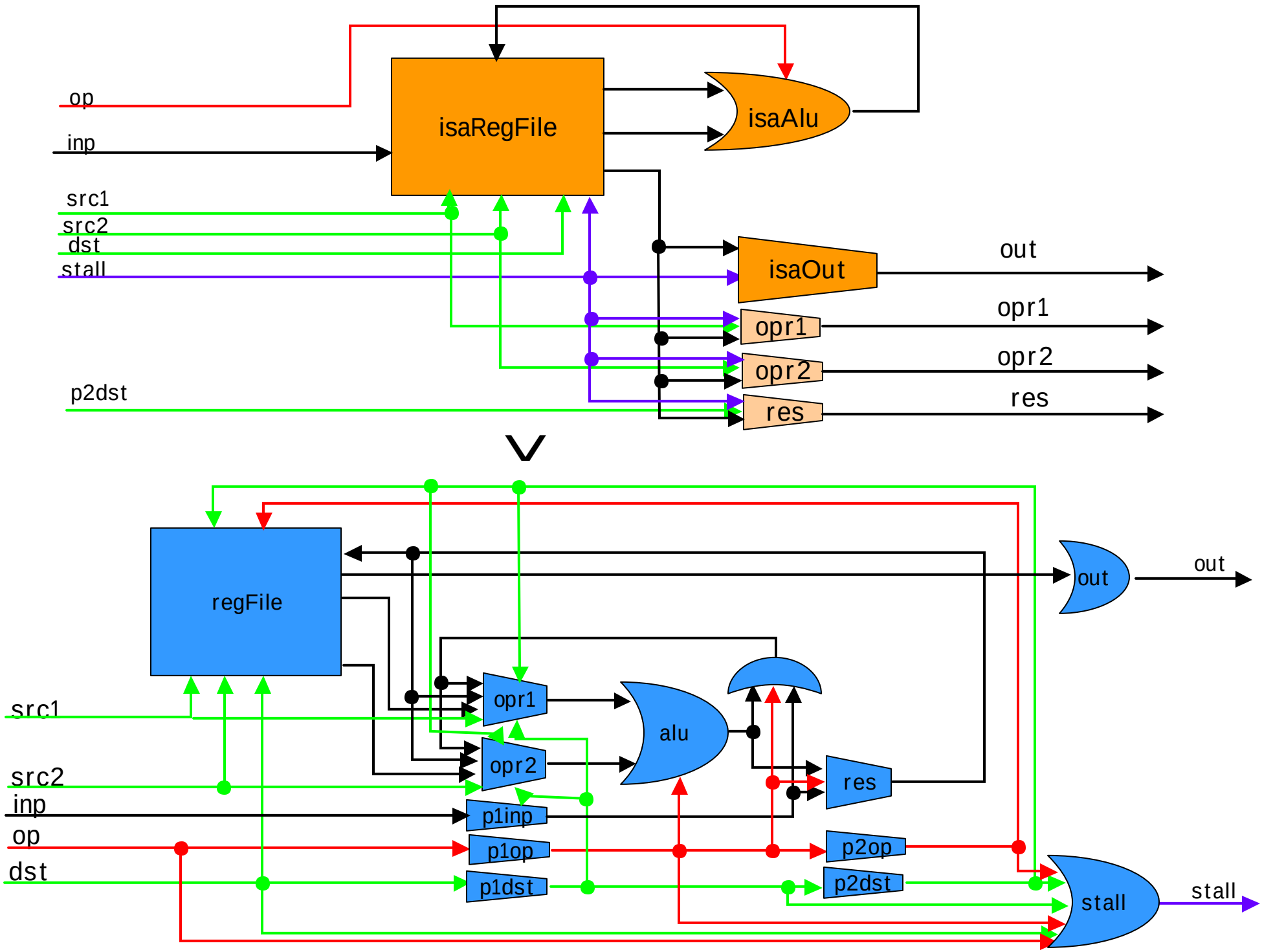

"opr1" proof
op
inp
isaRegFile
isaAlu
src1
src2
dst
stall
isaOut
out
opr1
opr2
res
p2dst
regFile
out
opr1
alu
opr2
res
p1inp
p1op
p1dst
p2op
p2dst
stall
stall


"opr1" proof
op
inp
isaRegFile
isaAlu
src1
src2
dst
stall
opr2
res
p2dst
opr2
res
regFile
opr1
alu
src1
src2
inp
op
dst
p1inp
p1op
p1dst
p2op
p2dst
stall
stall


"opr1" proof
op
inp
isaRegFile
isaAlu
src1
src2
dst
stall
opr1
opr2
res
opr1
opr2
res
p2dst
regFile
opr1
alu
src1
src2
inp
op
dst
p1inp
p1op
p1dst
p2op
p2dst
stall
stall


regFile
src1
src2
inp
op
dst
P1
FETCH


FETCH
EXECUTE
WRITE-BACK


op
inp
isaRegFile
isaAlu
src1
src2
dst
stall
isaOut
out
opr1
opr1
opr2
opr2
p2dst
res
res
regFile
out
out
opr1
alu
opr2
res
src1
src2
inp
p1inp
op
p1op
p2op
dst
p1dst
p2dst
stall
stall

But… is this really practical?

Outline

1 Informal Introduction

2 Formal Definitions

- Reactive Systems
- Witnessed Refinement Proofs
- Slicing Reactive Systems
- Decomposing Refinement Proofs

3 Formal Example: Three-Stage Pipeline

4 Informal Example: Dataflow Processor Array

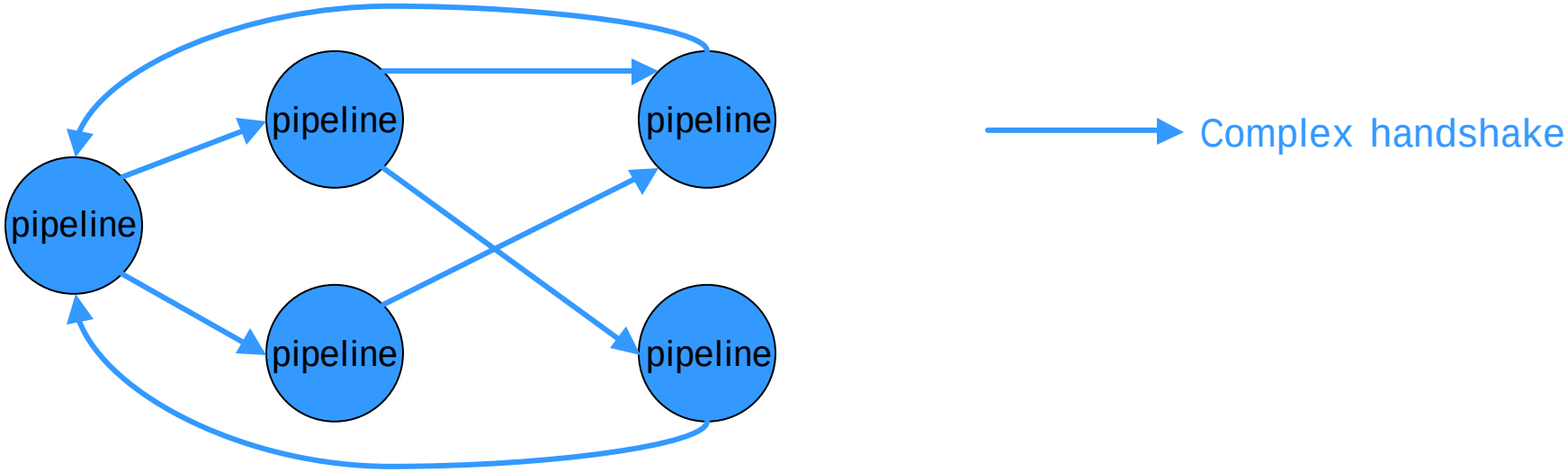
VGI Chip

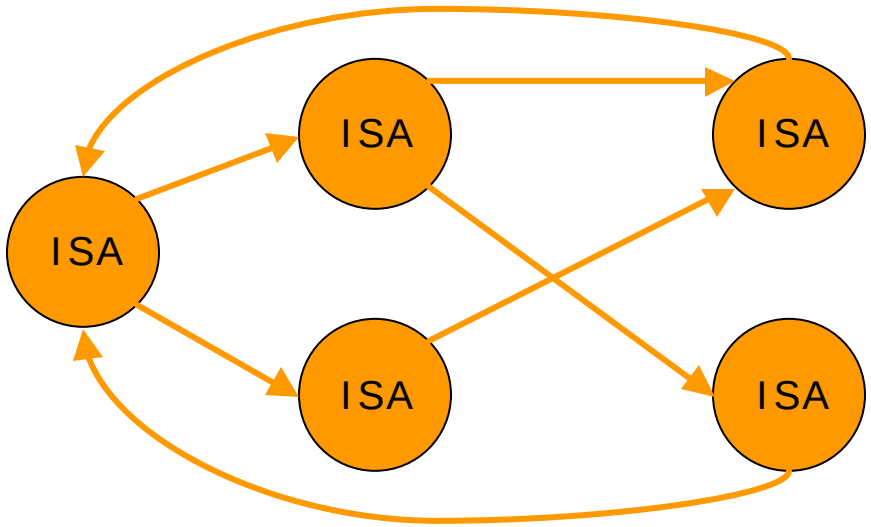
- VGI = “Video-Graphics-Image”
- Designed by Infopad group at Berkeley
- Purpose: web-based image processing
- Designed using
  - VHDL (control)
  - Schematics (data path)

VGI Architecture

- 16 clusters with 6 processors in each - 4 compute, 1 memory, 1 I/O
- ~30K logic gates per processor
- ~800 latches per processor
- Pipelined compute processors
- Low latency data transfer between processors - complex control

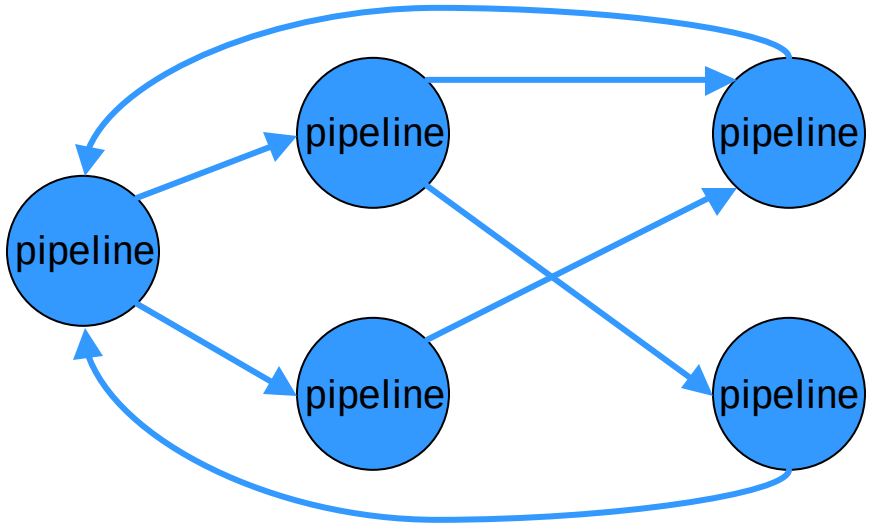
VGI Architecture





FIFO buffer

V



Complex handshake

Difficulties in Verification

Size of the VGI chip

- ~800 latches in each compute processor
- 64 compute processors
- Need two levels of “divide and conquer”

Different time scales

- Two-phase clk and level-sensitive latches in implementation
- Need special assume-guarantee rule

VGI Results

- All lemmas (exceptALU) checked by Mocha in a few minutes
- Three bugs in communication control found and fixed
- Abstract definitions crucial - designer insight needed

www.eecs.berkeley.edu/~mocha