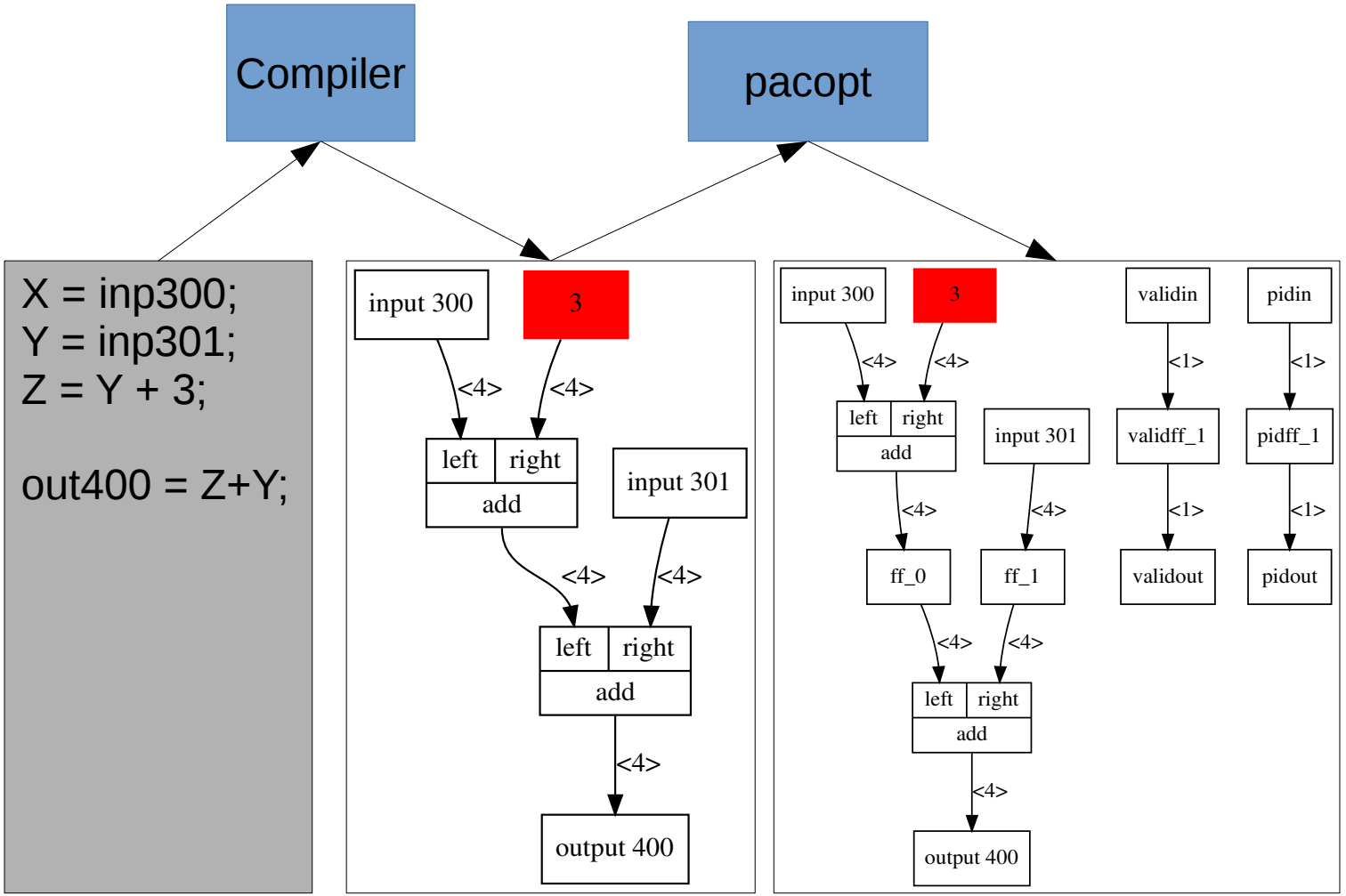
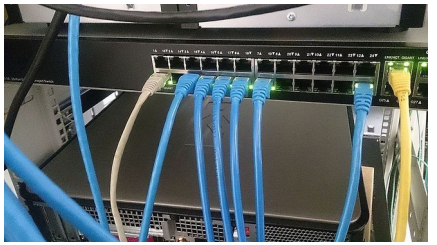
packetarc





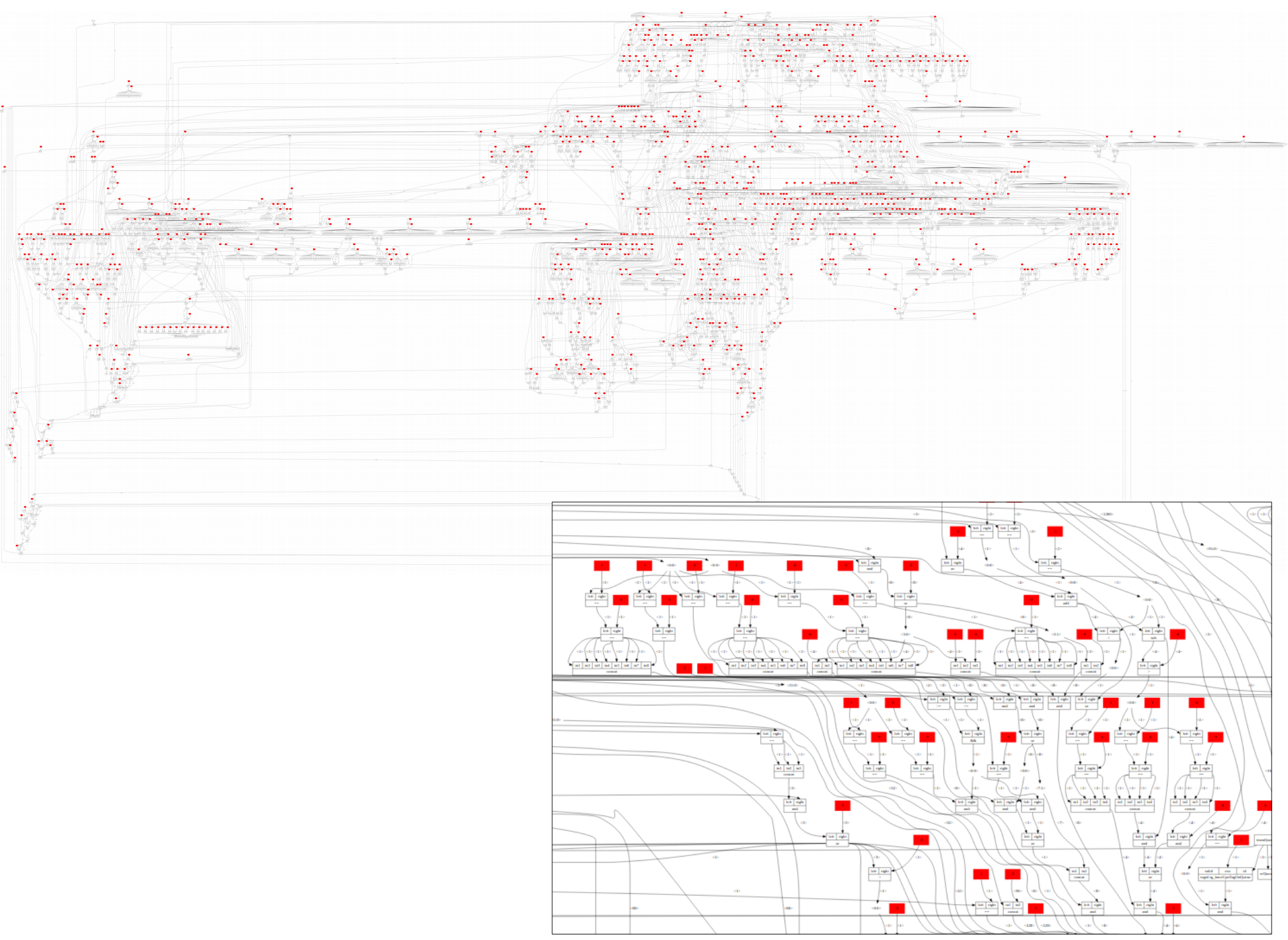
hello.c

hello.dot

hello_pacopt.dot

.dot example

```
digraph packetarc {
node [shape=record];
in_1 [label="input 300"];
in_2 [label="input 301"];
c_3 [label="3" shape=plaintext color="red" style="filled"];
opadd_4 [label="{{<left> left | <right> right} | <out> add}" debug="25454 1"];
opadd_5 [label="{{<left> left | <right> right} | <out> add}" debug="25454 1"];
out_6 [label="output 400"];
in_1->opadd_4:left[label="<4>"];
c_3->opadd_4:right[label="<4>"];
opadd_4:out->opadd_5:left[label="<4>"];
in_2->opadd_5:right[label="<4>"];
opadd_5:out->out_6 [label="<4>"];
}
```

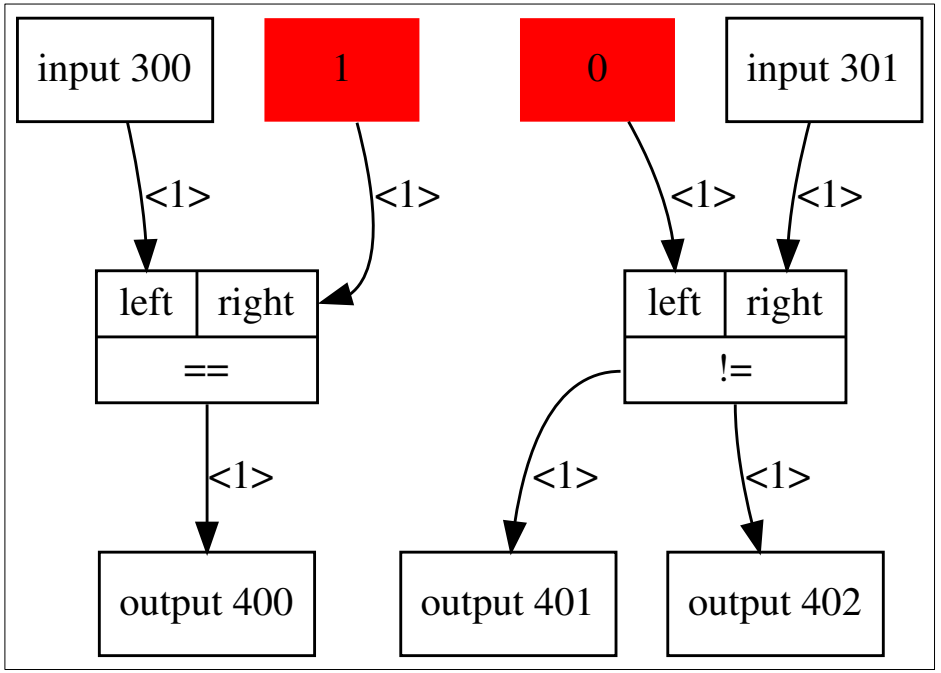


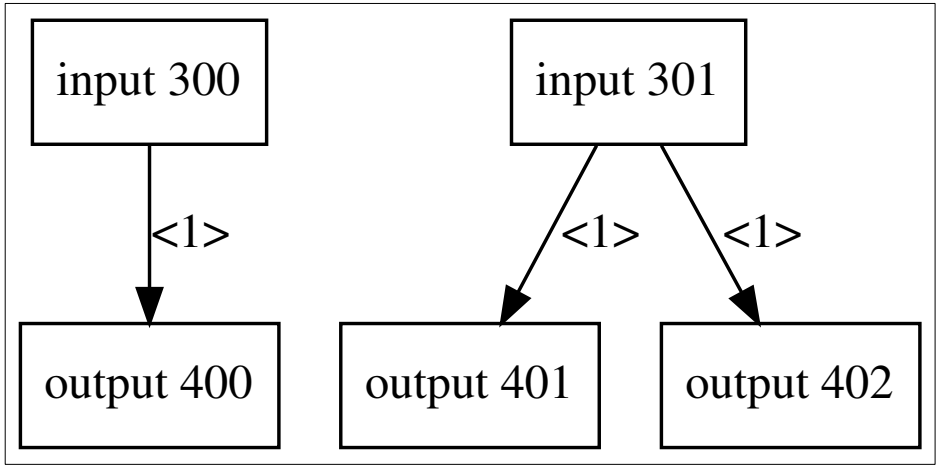
Optimization

- eq_1
- bitwidth
- remove_duplicates
- const_merging
- tree_height_reduction

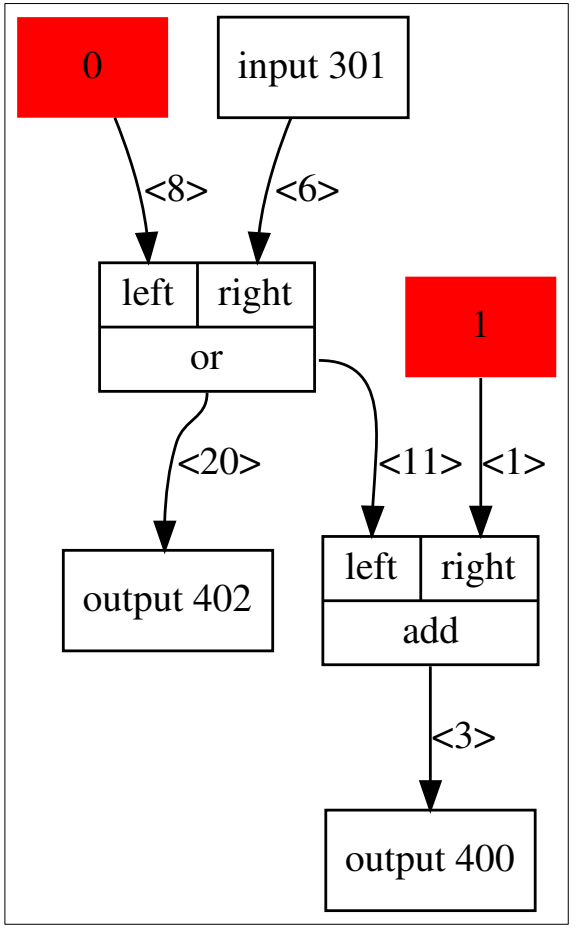
- operator optimization
- algebraic simplification

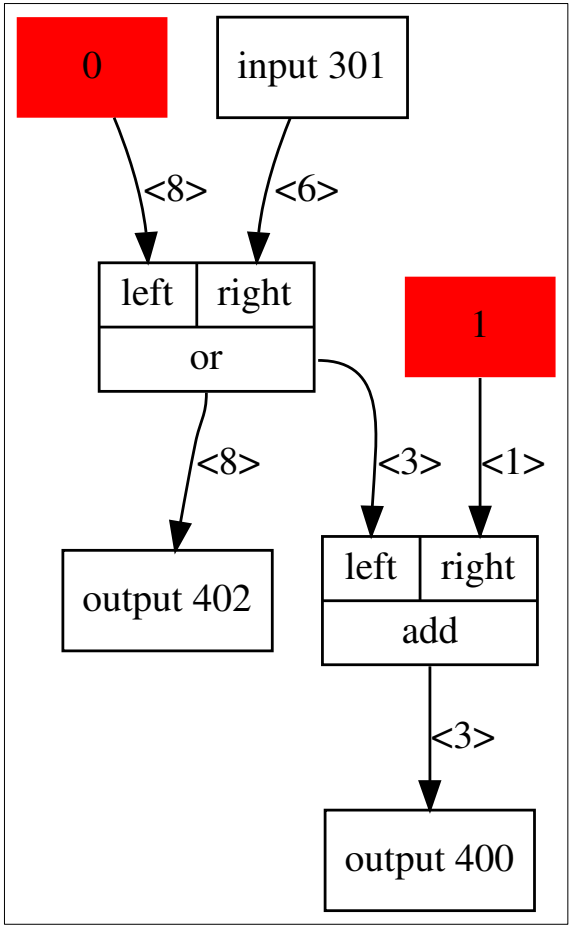
eq_1


input 300
1
0
input 301
<1>
<1>
<1>
<1>
left
right
==
left
right
!=
<1>
<1>
<1>
output 400
output 401
output 402

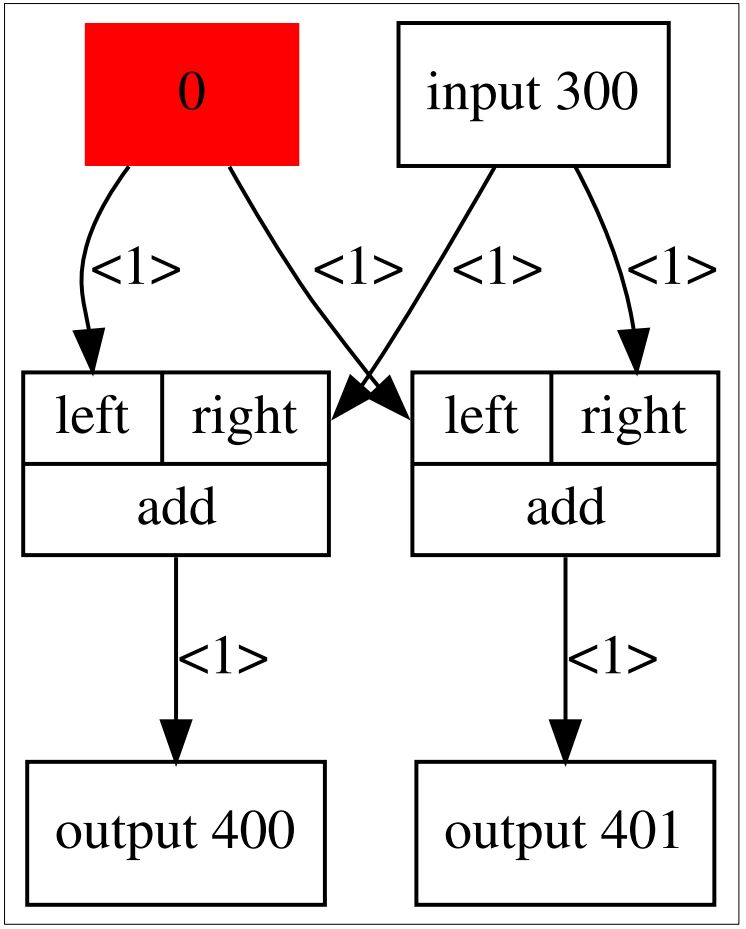

input 300
input 301
<1>
<1>
<1>
output 400
output 401
output 402

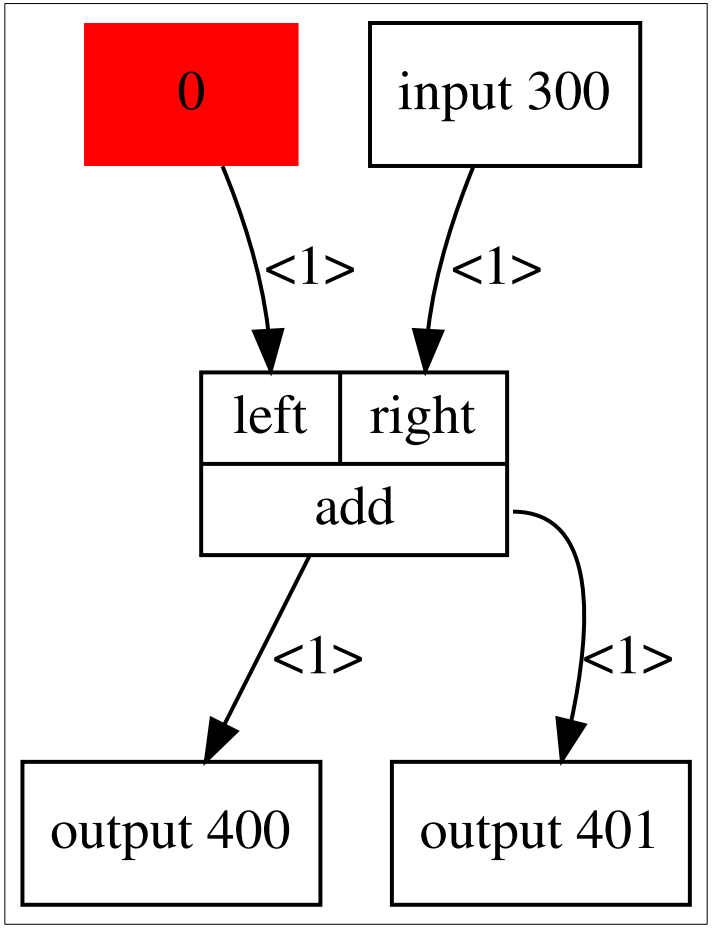
bitwidth


0
input 301
<8>
<6>
left
right
or
1
<20>
<11>
<1>
output 402
left
right
add
<3>
output 400

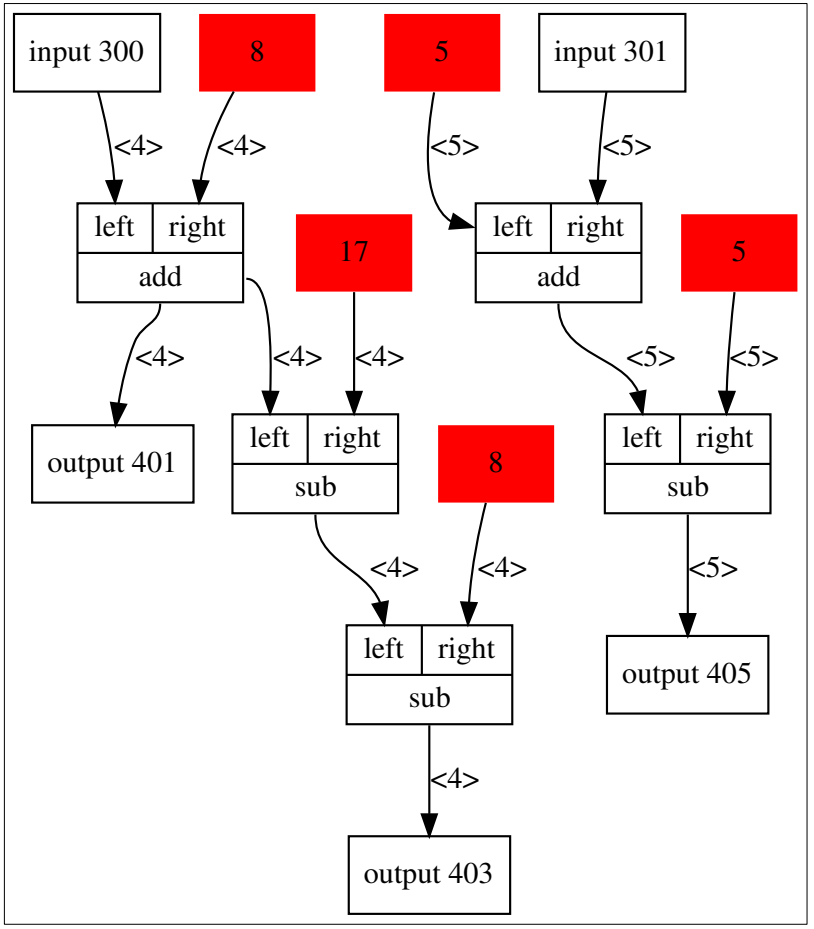

0
input 301
<8>
<6>
left
right
or
1
<8>
<3>
<1>
output 402
left
right
add
<3>
output 400

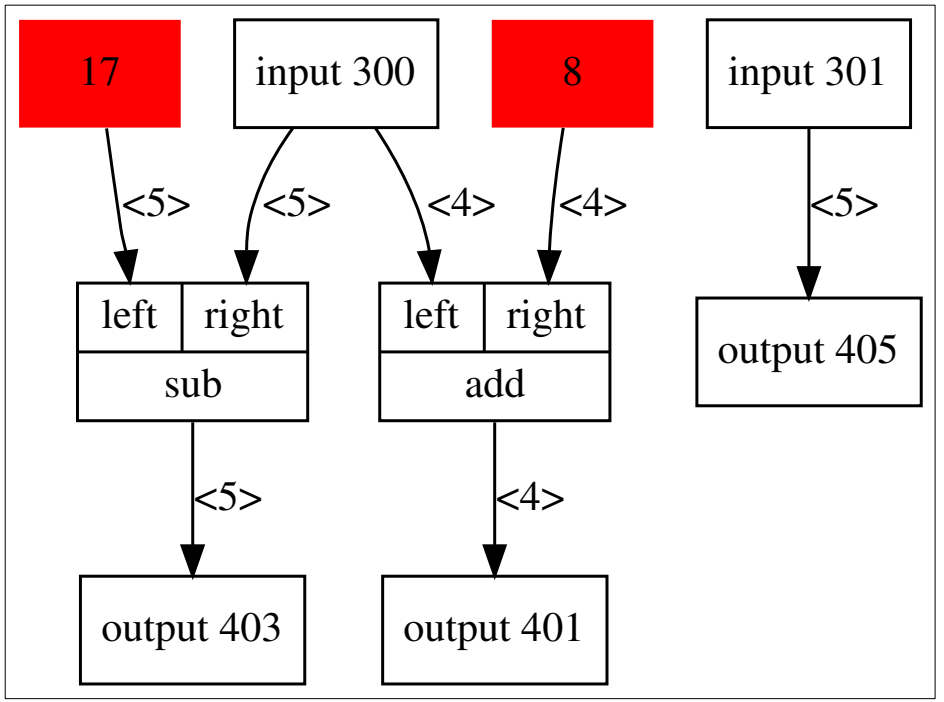
remove_duplicates



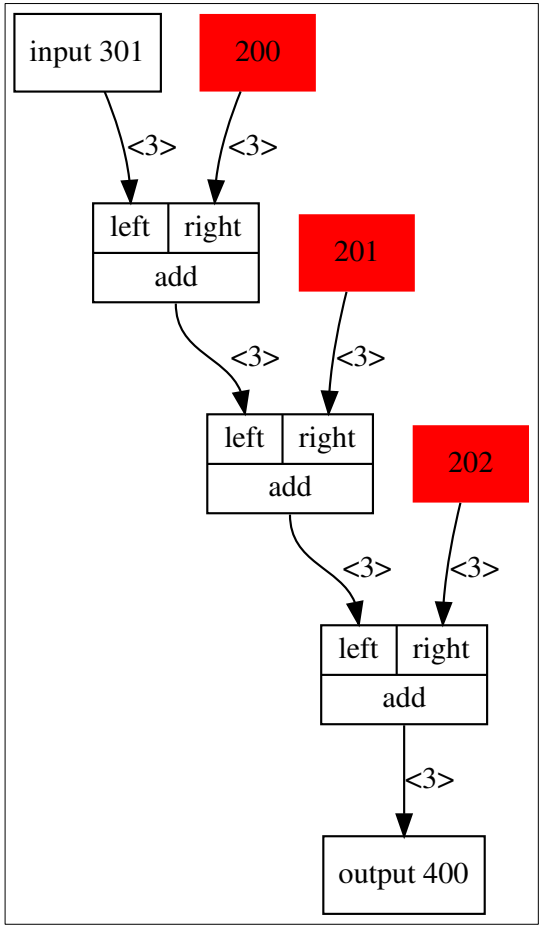


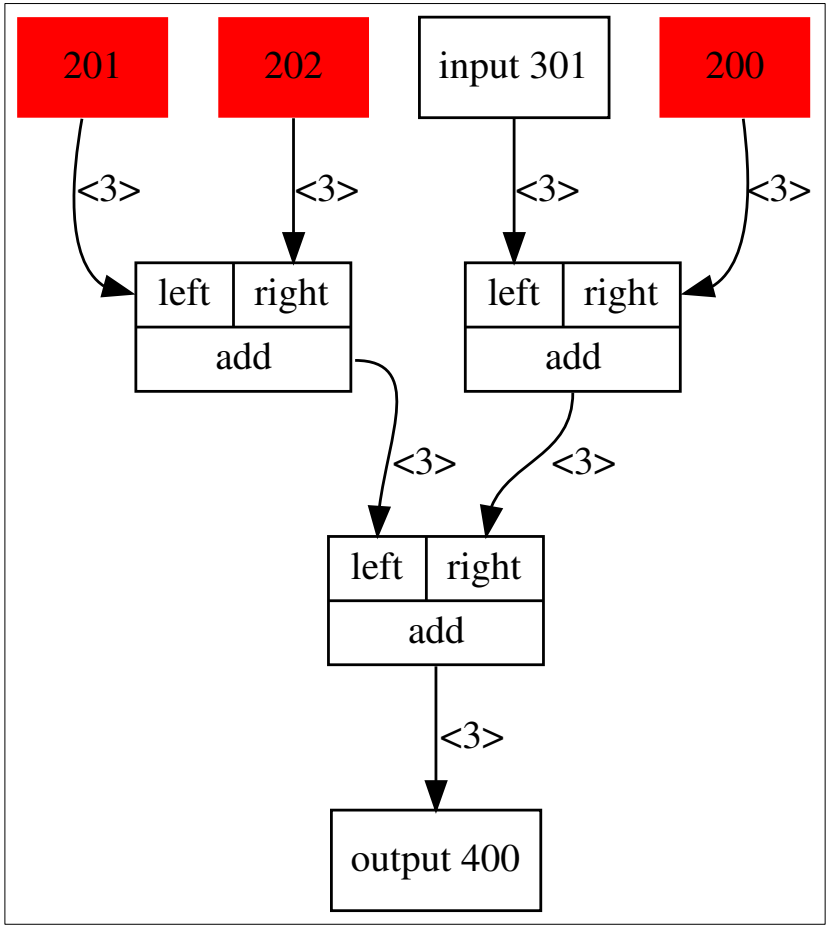
const_merging





tree_height_reduction





Unlikely cases

- operator optimization 4*x → x<<2
- algebraic simplification x=a*b+a*c → x=a*(b+c)

Stats!

	target delay = 80			target delay = 30		
	opdelay_add	tag_6x10G	c1mepp	opdelay_add	tag_6x10G	c1mepp
eq_1	0,000	0,002	0,001	0,000	0,001	−0,035
bitwidth	0,309	0,000	0,001	0,444	0,000	0,000
const_merging	0,489	0,016	0,000	0,258	0,006	0,000
remove_duplicates	0,000	0,000	----	0,000	0,002	----
bitwidth & tree height reduction	0,628	0,000	0,001	0,455	−0,010	0,008
all	0,665	0,002	----	0,643	−0,009	----
best	0,665	0,018	0,001	0,643	0,006	0,008

Lessons learned

- tools
- time spent understanding the structure
- time spent on I/O
- benchmarking is hard!
- bit-logic is hard!

Questions?

Questions?

Thanks to packet architects

and thank you for listening