The University of Manchester

Teak optimisation language

Some of Teak's network optimisations are hand coded but most are described in an optimisation language and can be read by the tool at runtime.

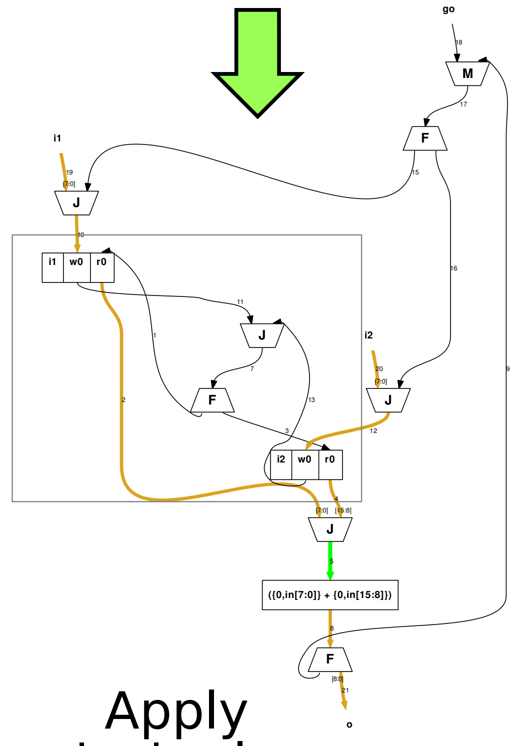
For example, Variable (V) components which are written in parallel can be combined. This rule is called 'vv' and is often useful for joining procedure inputs removing Forks and Joins.

```
rule vv "merge Vs with joined writes"
    from
        j1: J ([...,wd1,...,wd2,...], wd)
        v1: V ([wg1], [wd1], rg1, rd1)
        v2: V ([wg2], [wd2], rg2, rd2)
    when
        fullWidthWrite v1 wg1 &&
        fullWidthWrite v2 wg2
    to
        j1: J ([wd',...,...,...], wd)
        j2: J ([wg1,wg2], w)
        v: V { name = newName, width = newWidth }
            ([w], [wd'], [rg1,rg2], [rd1,rd2'])
    where
        newName = v1.name ++ "-" ++ v2.name
        newWidth = v1.width + v2.width
        w = link { width = wg1.width + wg2.width,
            wOffset = 0 }
        wd' = link { width = 0 }
        rd2' = rd2 { rOffset = rOffset + v1.width }
```

```
Match 1/1:
from
  j1: C7 (TeakJ) [[L11,L13],L7]
  v1: C5 (TeakV "i1" 8 [] [0] [0]) [[L10],[L11],[L1],[L2]]
  v2: C6 (TeakV "i2" 8 [] [0] [0]) [[L12],[L13],[L3],[L4]]
to
  j1: C7 (TeakJ) [[L-2],L
7]
  j2: C0 (TeakJ) [[L10,L12],L-1]
  v:  C0 (TeakV "i1-i2" 16 [] [0] [0,8])
    [[L-1],[L-2],[L1,L3],[L2,L4]]
where
  j1/1:      []
  j1/2:      []
  j1/3:      []
  rd1:       [L2{rOffset = 0}]
  rd2:       [L4{rOffset = 0}]
  rg1:       [L1]
  rg2:       [L3]
  wd:        L7
  wd1:       L11
  wd2:       L13
  wg1:       L10{wOffset = 0}
  wg2:       L12{wOffset = 0}
  j1:        C7 (TeakJ) [[L11,L13],L7]
  v1:        C5 (TeakV "i1" 8 [] [0] [0]) [[L10],[L11],[L1],[L2]]
  v2:        C6 (TeakV "i2" 8 [] [0] [0]) [[L12],[L13],[L3],[L4]]
  w:         L-1{wOffset = 0}
  wd':       L-2
  newName:   "i1-i2"
  newWidth:  16
  rd2':      [L4{rOffset = 8}]
new links
  -1: NetlistLink -1 16
  -2: NetlistLink -2 0
```

```
procedure add (
    input i1, i2 : 8 bits;
    output o : 9 bits
) is
begin
    loop
        i1, i2 -> then
            o <- i1 + i2
        end
    end
end
```

Without optimisation



Apply 'vv' rule