Generic parallel scan

Conal Elliott

Target Data Sciences

October 5, 2016 (revised May 2017)

Some applications of parallel scan

From a longer list in *Prefix Sums and Their Applications*:

- Adding multi-precision numbers
- Polynomial evaluation
- Solving recurrences
- Sorting
- Solving tridiagonal linear systems
- Lexical analysis
- Regular expression search

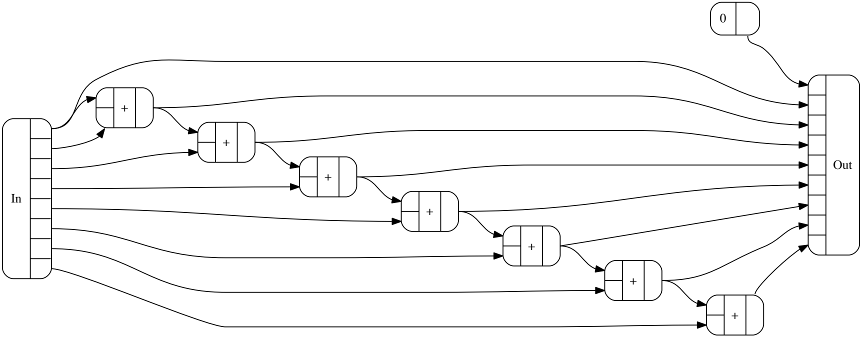
Prefix sum (left scan)

Given $a_1, \ldots, a_n$, compute

$$b_k = \sum_{1 \leqslant i < k} a_i \qquad \text{for } k = 1, \ldots, n+1$$

Note that a_k does *not* influence b_k.

Linear left scan



Work: $O(n)$

Depth: $O(n)$ (ideal parallel "time")

Linear *dependency chain* thwarts parallelism (depth < work).

Scan class

$$\textbf{class}\ \mathit{Functor}\ f \Rightarrow \mathit{LScan}\ f\ \textbf{where}$$
$$\quad \mathit{lscan} :: \mathit{Monoid}\ a \Rightarrow f\ a \rightarrow f\ a \times a$$

Specification (if *Traversable* f):

$$\mathit{lscan} \equiv \mathit{swap} \circ \mathit{mapAccumL}\ (\lambda \mathit{acc}\ a \rightarrow (\mathit{acc} \diamond a, \mathit{acc}))\ \varepsilon$$

Generic building blocks

$$\begin{array}{llll}
\textbf{data} & V_1 & a & \text{-- void} \\
\textbf{newtype} & U_1 & a = U_1 & \text{-- unit} \\
\textbf{newtype} & Par_1 & a = Par_1\ a & \text{-- singleton} \\
\textbf{data} & (f + g) & a = L_1\ (f\ a) \mid R_1\ (g\ a) & \text{-- sum} \\
\textbf{data} & (f \times g) & a = f\ a \times g\ a & \text{-- product} \\
\textbf{newtype} & (g \circ f) & a = Comp_1\ (g\ (f\ a)) & \text{-- composition}
\end{array}$$

Plan:

- Define parallel scan for each.
- Use directly, *or*
- automatically via (derived) encodings.

Easy instances

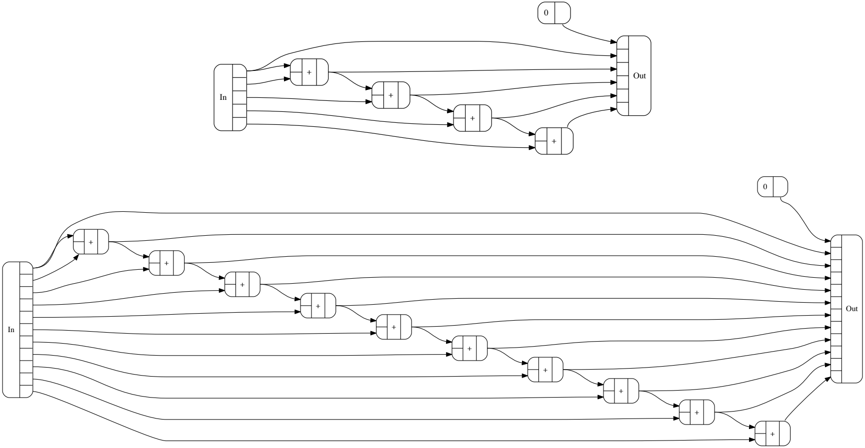
$\textbf{instance}\ \mathit{LScan}\ V_1\quad \textbf{where}\ \mathit{lscan} = \lambda\ \textbf{case}$

$\textbf{instance}\ \mathit{LScan}\ U_1\quad \textbf{where}\ \mathit{lscan}\ U_1 = (U_1, \varepsilon)$

$\textbf{instance}\ \mathit{LScan}\ \mathit{Par}_1\ \textbf{where}\ \mathit{lscan}\ (\mathit{Par}_1\ a) = (\mathit{Par}_1\ \varepsilon, a)$

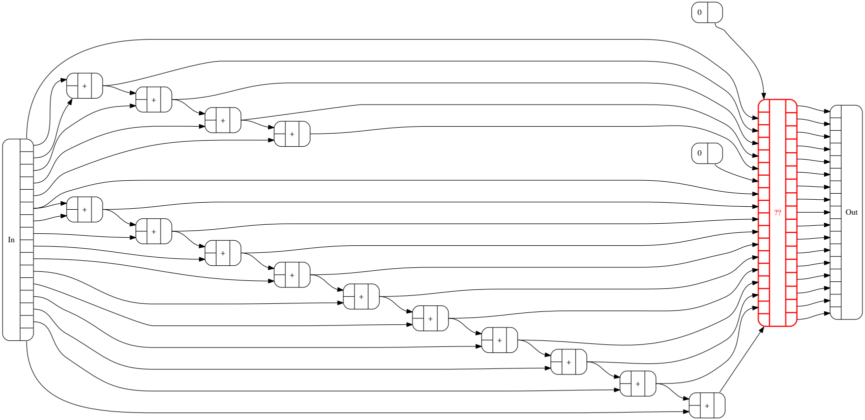
$$
\begin{aligned}
&\textbf{instance}\ (\mathit{LScan}\ f, \mathit{LScan}\ g) \Rightarrow \mathit{LScan}\ (f + g)\ \textbf{where}\\
&\quad \mathit{lscan}\ (L_1\ \mathit{fa}) = \mathit{first}\ L_1\ (\mathit{lscan}\ \mathit{fa})\\
&\quad \mathit{lscan}\ (R_1\ \mathit{ga}) = \mathit{first}\ R_1\ (\mathit{lscan}\ \mathit{ga})
\end{aligned}
$$

Product example: *LVec* 5 × *LVec* 11

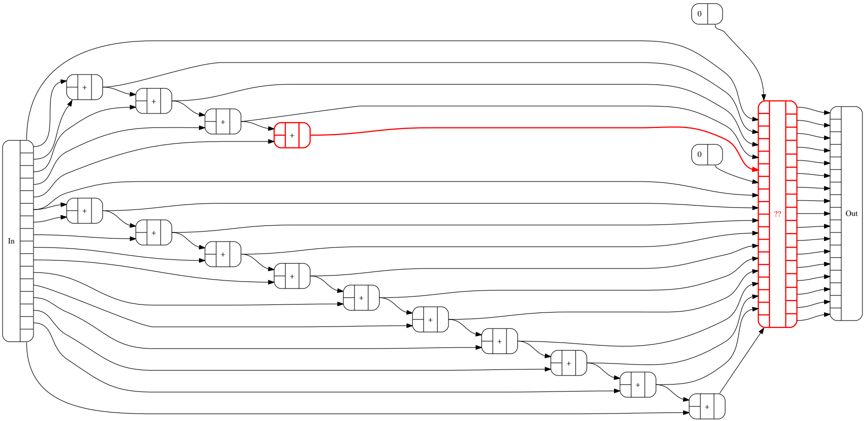


Then what?

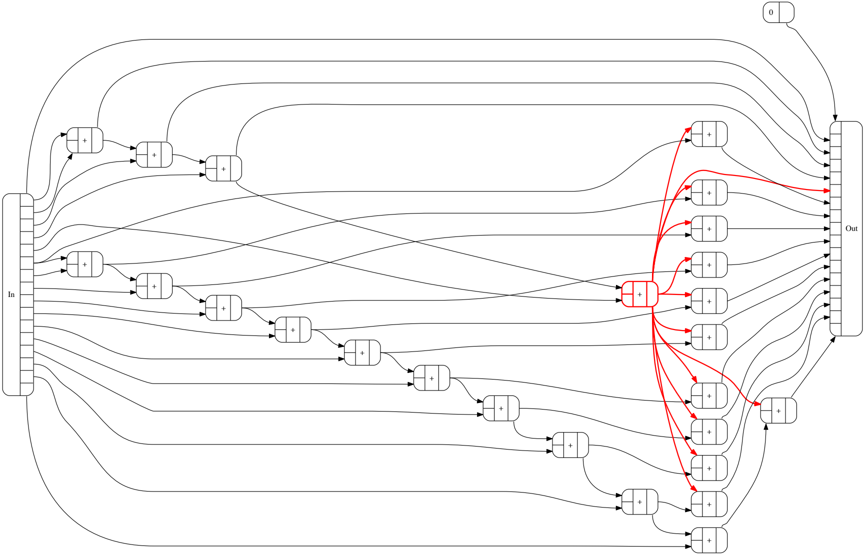
Combine?



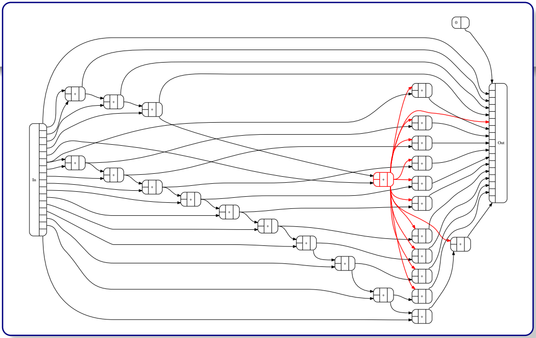
Combine?



Right adjustment



Products



$$\textbf{instance}\ (LScan\ f, LScan\ g) \Rightarrow LScan\ (f \times g)\ \textbf{where}$$
$$\quad lscan\ (fa \times ga) = (fa' \times ((fx \diamond\ ) \mathbin{<\$>} ga'), fx \diamond gx)$$
$$\quad\quad \textbf{where}$$
$$\quad\quad\quad (fa', fx) = lscan\ fa$$
$$\quad\quad\quad (ga', gx) = lscan\ ga$$

Vector type families

Right-associated:

$$
\begin{array}{l}
\textbf{type family } \mathit{RVec}\ n \textbf{ where} \\
\quad \mathit{RVec}\ Z \quad\quad = U_1 \\
\quad \mathit{RVec}\ (S\ n) = \mathit{Par}_1 \times \mathit{RVec}\ n
\end{array}
$$

Left-associated:

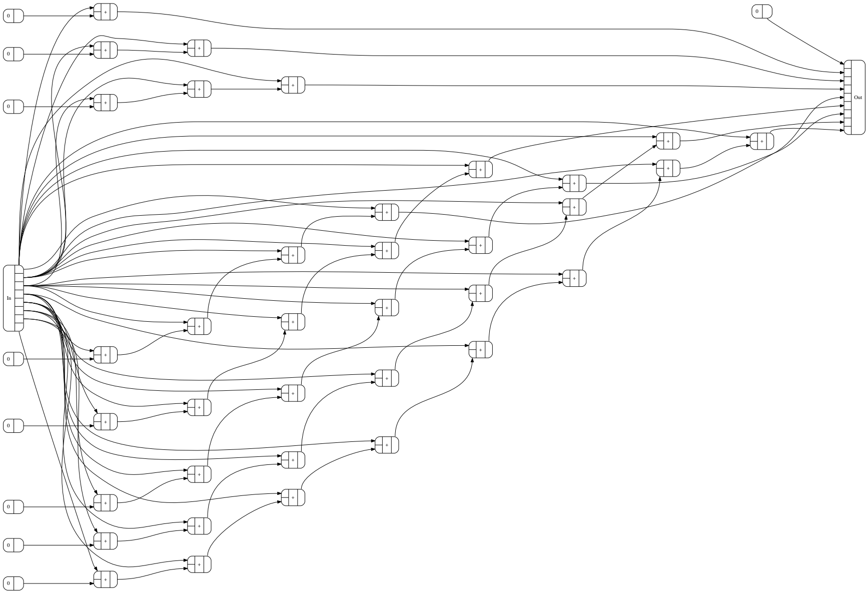
$$
\begin{array}{l}
\textbf{type family } \mathit{LVec}\ n \textbf{ where} \\
\quad \mathit{LVec}\ Z \quad\quad = U_1 \\
\quad \mathit{LVec}\ (S\ n) = \mathit{LVec}\ n \times \mathit{Par}_1
\end{array}
$$

Also convenient:

$\textbf{type } \mathit{Pair} = \mathit{Par}_1 \times \mathit{Par}_1$ -- or $\mathit{RVec}\ 2$ or $\mathit{LVec}\ 2$

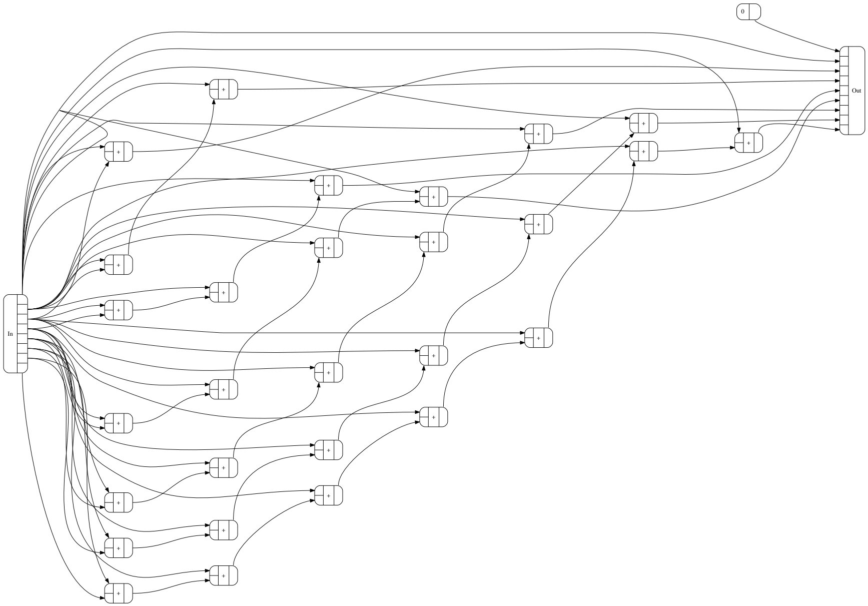
RVec 8 (unoptimized)

work: 36, depth: 8



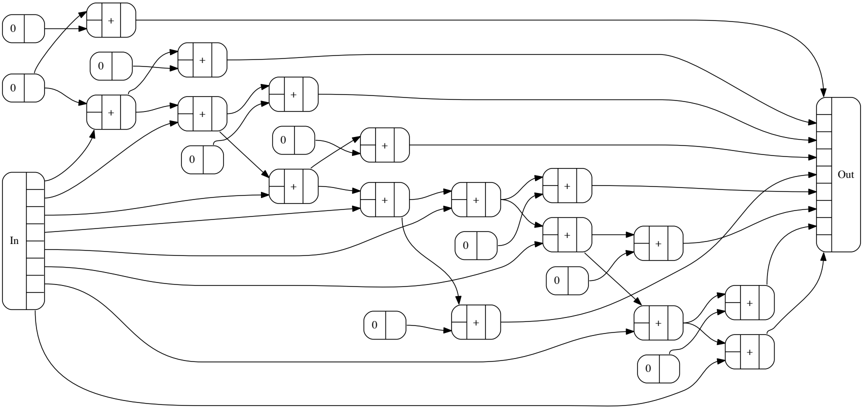
RVec 8 (optimized)

work: 28, depth: 7



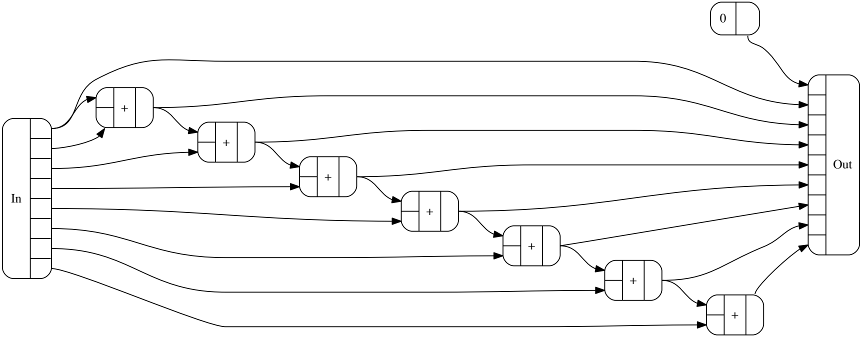
LVec 8 (unoptimized)

work: 16, depth: 8



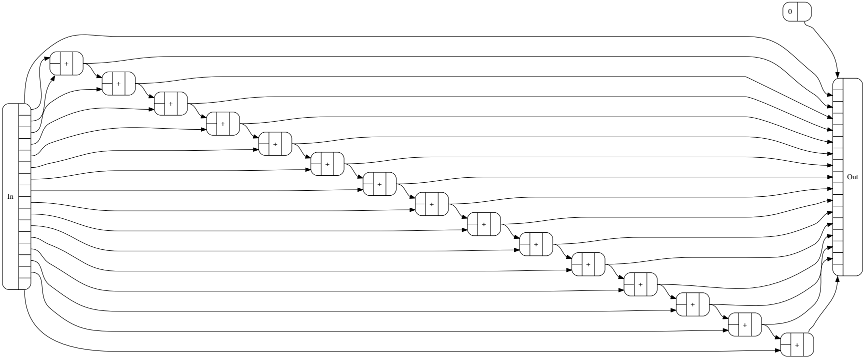
LVec 8 (optimized)

work: 7, depth: 7



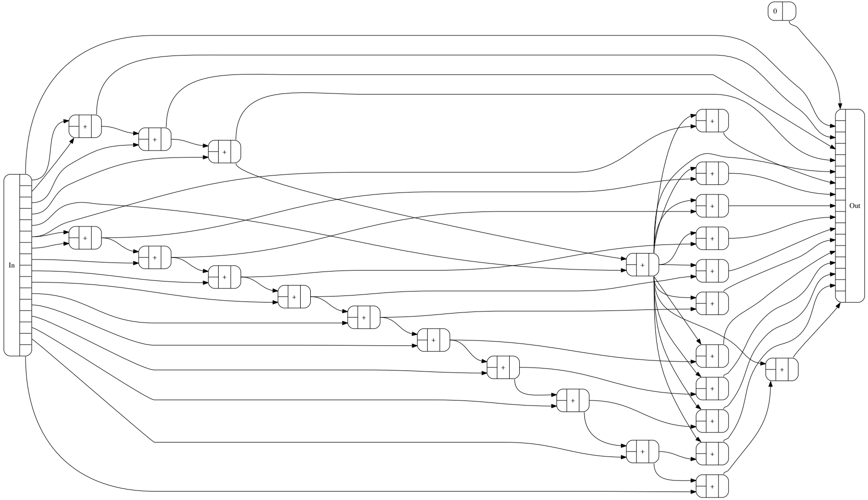
LVec 16 (optimized)

work: 15, depth: 15



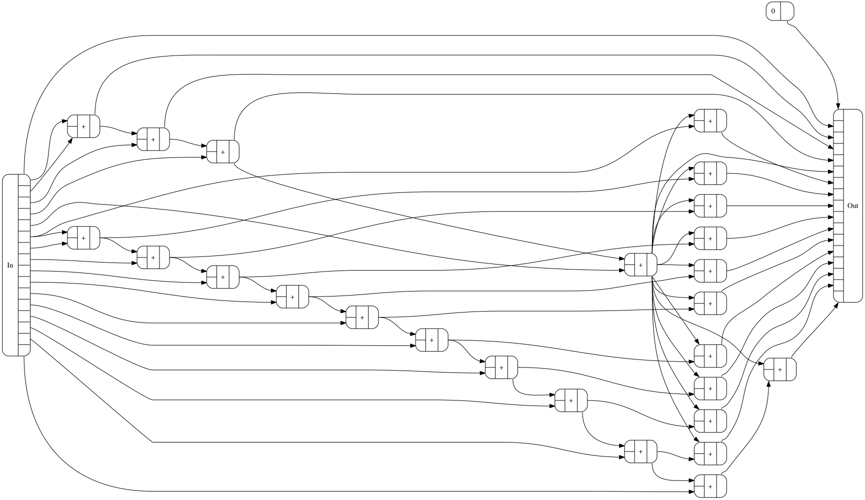
LVec 5 × *LVec* 11

work: 25, depth: 11



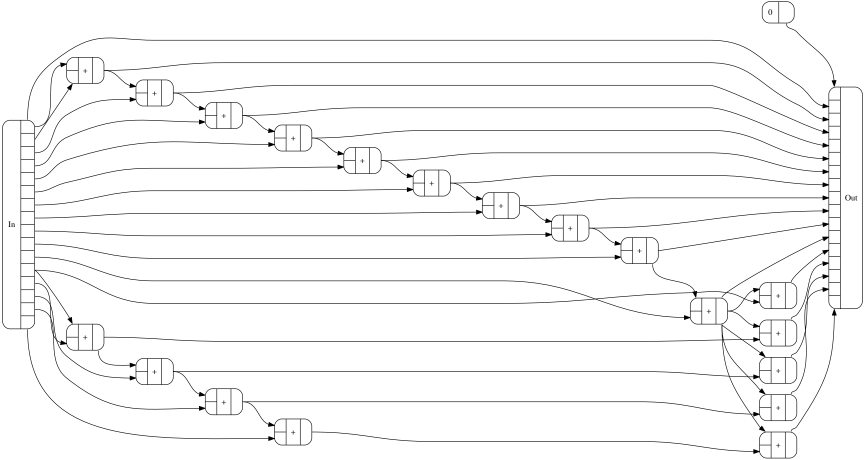
$5 + 11$

work: 25, depth: 11



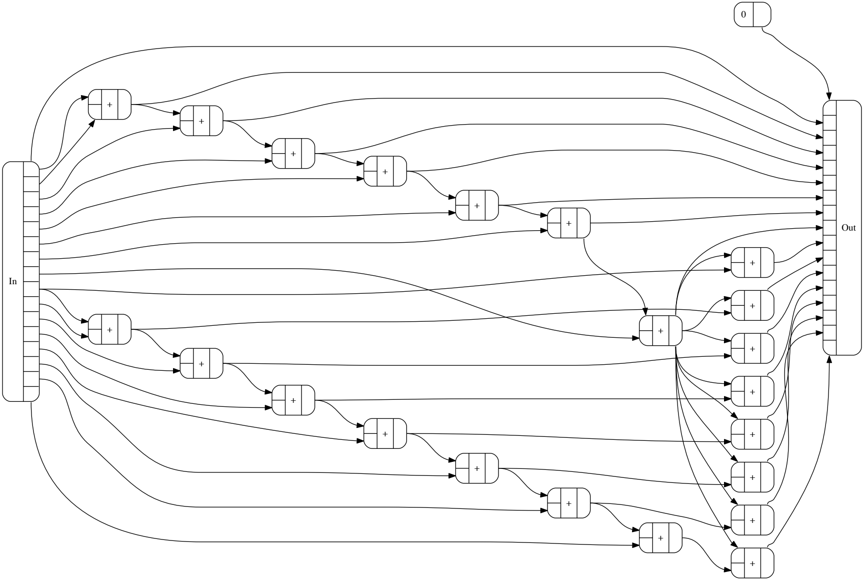
$11 + 5$

work: 19, depth: 11

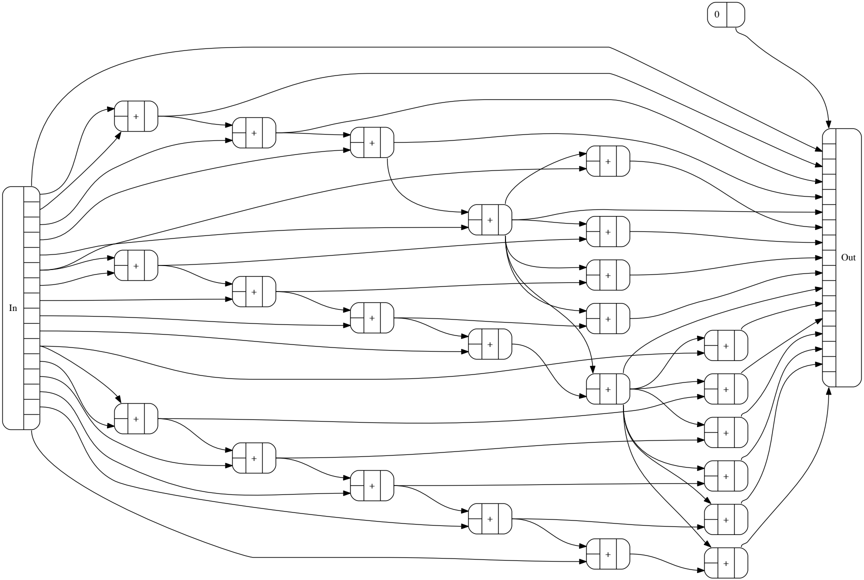


$8 + 8$

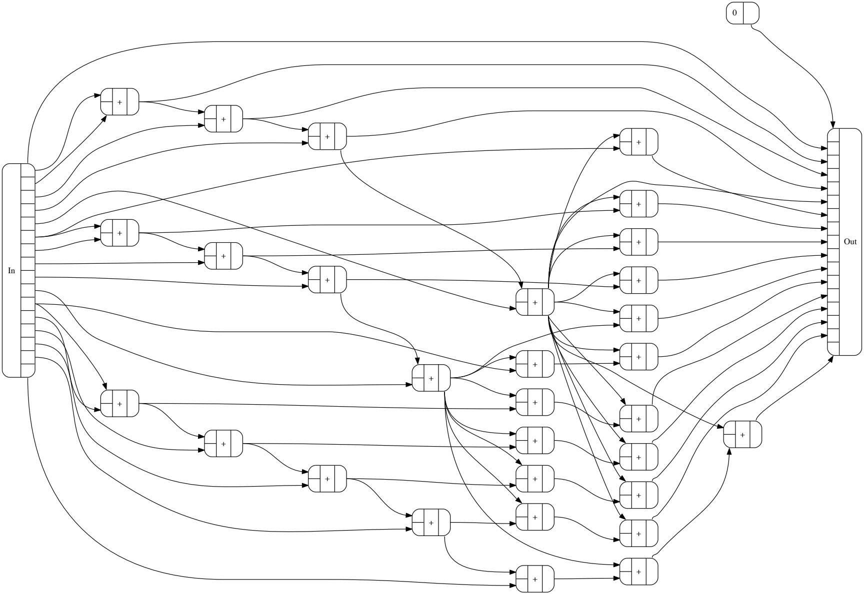
work: 22, depth: 8



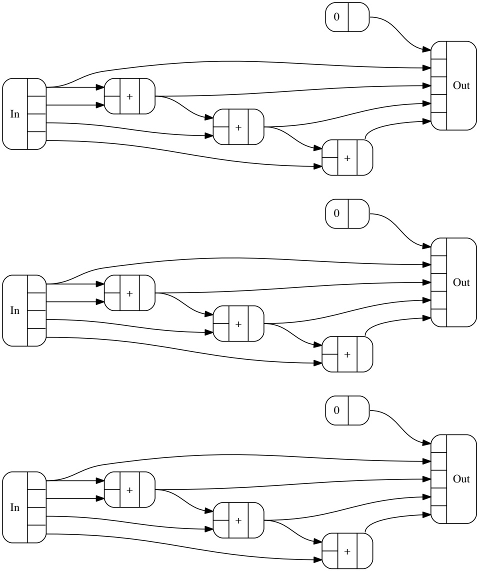
$(5 + 5) + 6$ work: 24, depth: 6



$5 + (5 + 6)$ work: 30, depth: 7

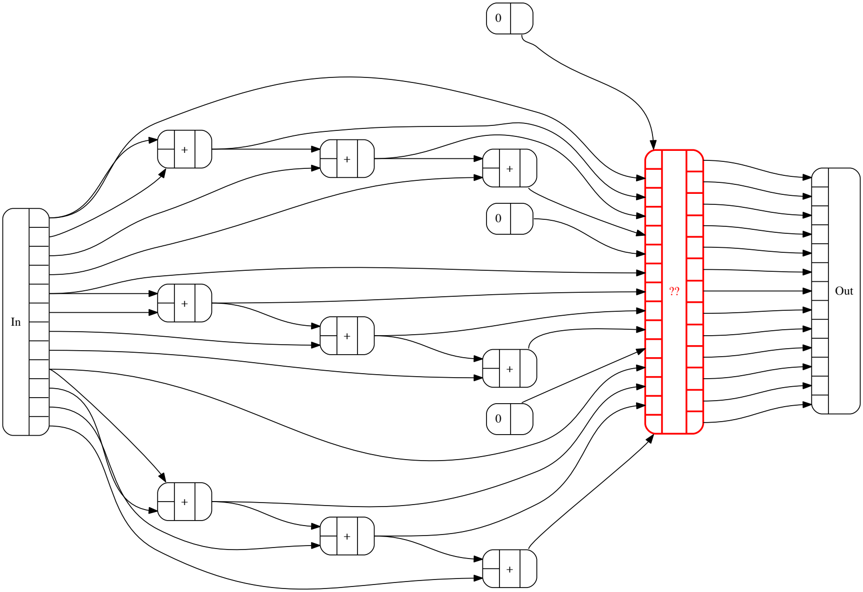


Composition example: $LVec\ 3 \circ LVec\ 4$

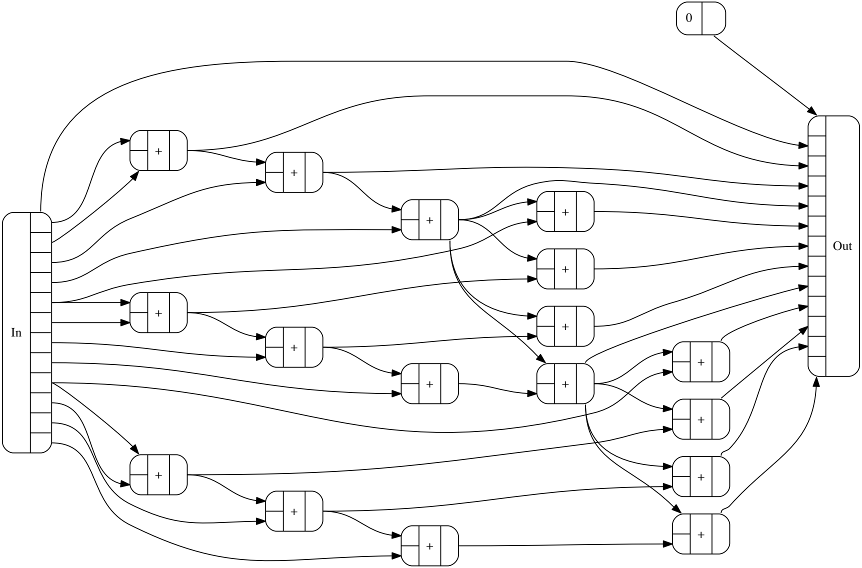


Then what?

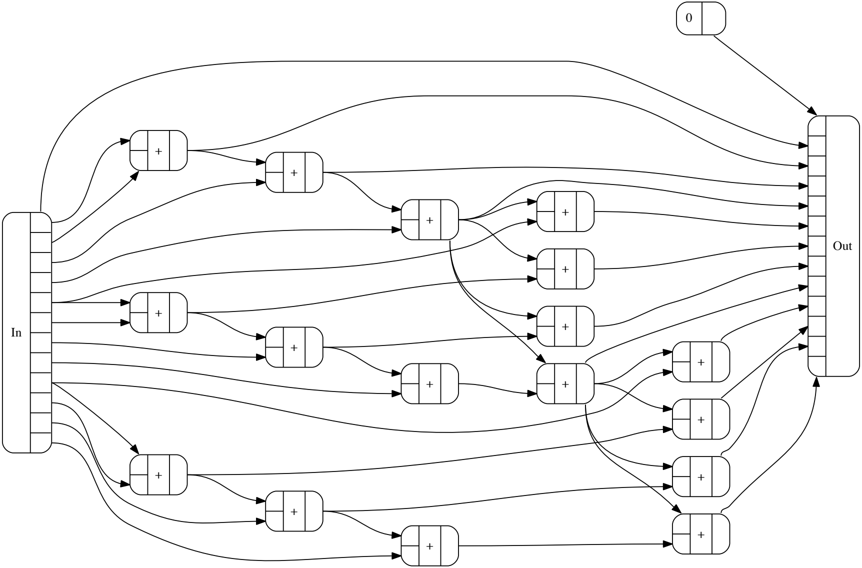
Combine?



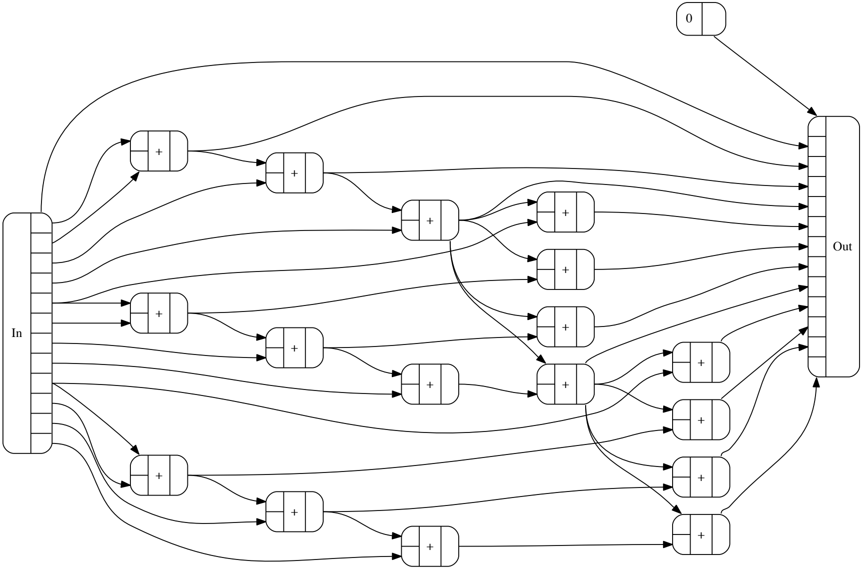
$(4 + 4) + 4$



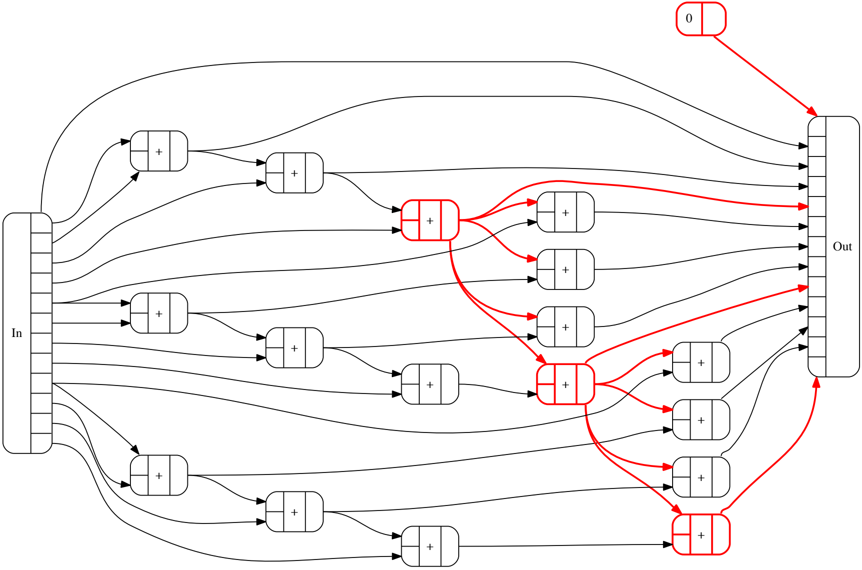
3×4



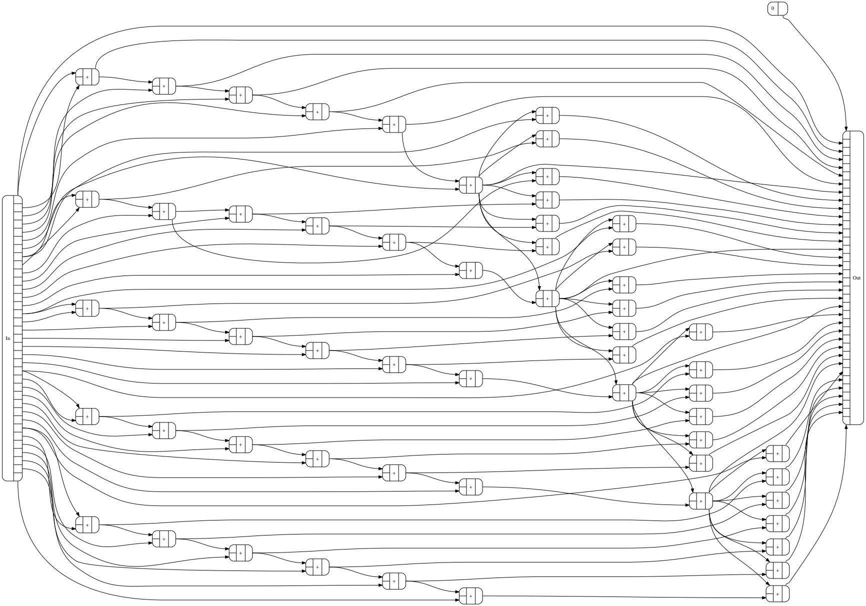
$LVec\ 3 \circ LVec\ 4$



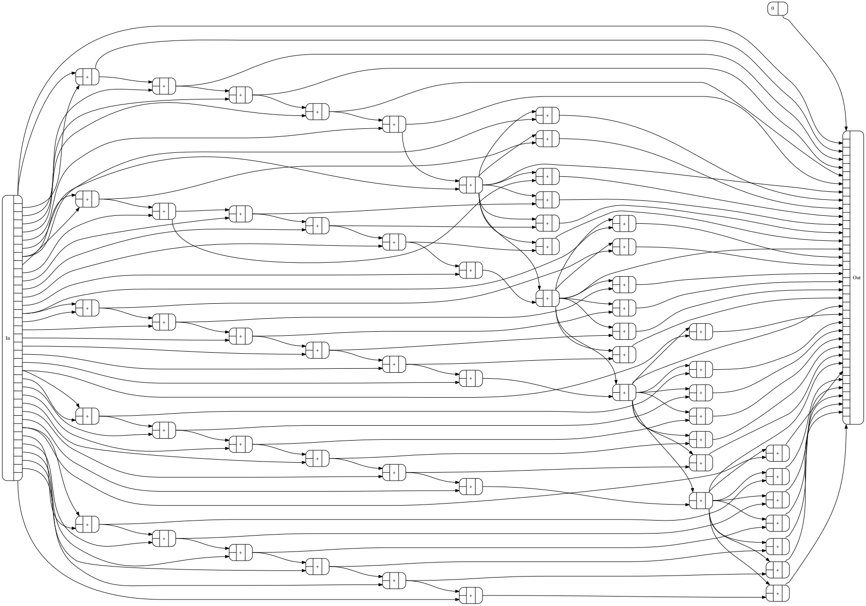
LVec 3 ∘ *LVec* 4



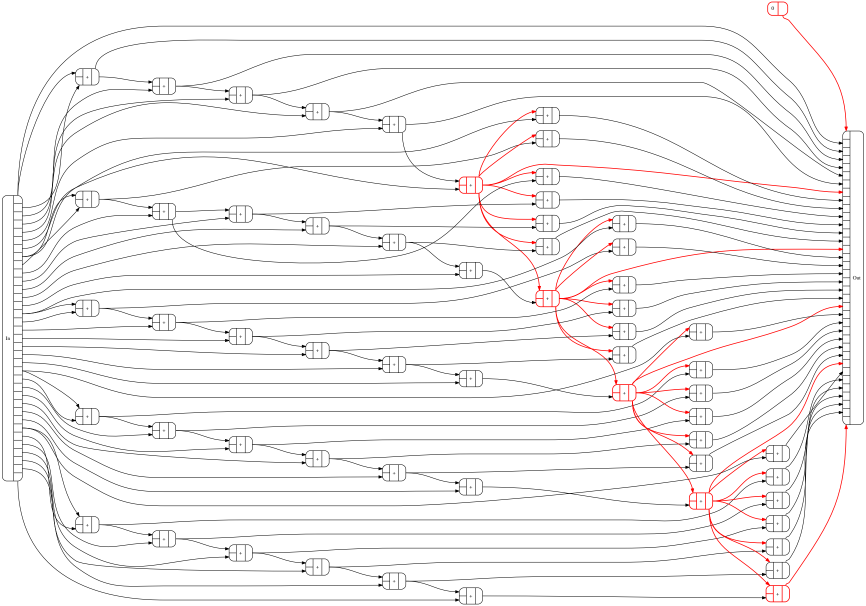
$(((7+7)+7)+7)+7$



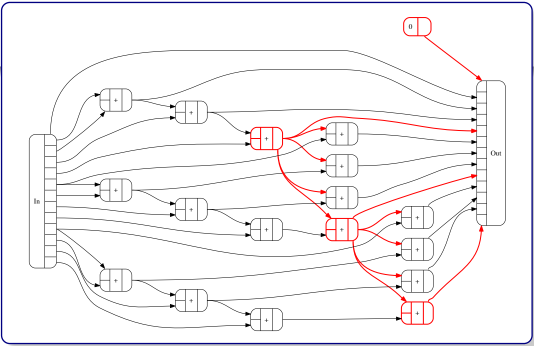
LVec 5 ∘ *LVec* 7



LVec 5 ∘ *LVec* 7



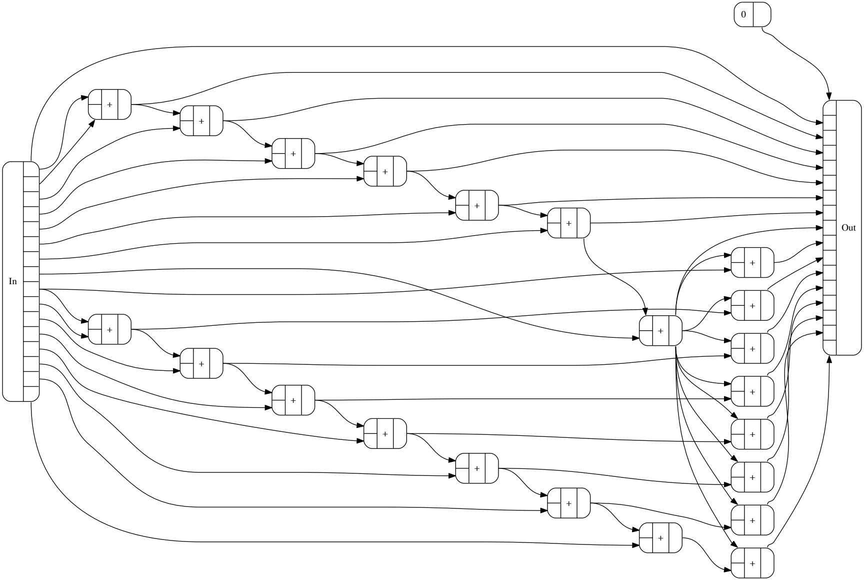
Composition



$$
\begin{array}{l}
\textbf{instance}\ (\mathit{LScan}\ g, \mathit{LScan}\ f, \mathit{Zip}\ g) \Rightarrow \mathit{LScan}\ (g \circ f)\ \textbf{where} \\
\quad \mathit{lscan}\ (\mathit{Comp}_1\ \mathit{gfa}) = (\mathit{Comp}_1\ (\mathit{zipWith}\ \mathit{adjustl}\ \mathit{tots}'\ \mathit{gfa}'), \mathit{tot}) \\
\qquad \textbf{where} \\
\qquad\quad (\mathit{gfa}', \mathit{tots}) = \mathit{unzip}\ (\mathit{lscan} \mathbin{<\$>} \mathit{gfa}) \\
\qquad\quad (\mathit{tots}', \mathit{tot}) = \mathit{lscan}\ \mathit{tots} \\
\qquad\quad \mathit{adjustl}\ t \quad = \mathit{fmap}\ (t \diamond)
\end{array}
$$

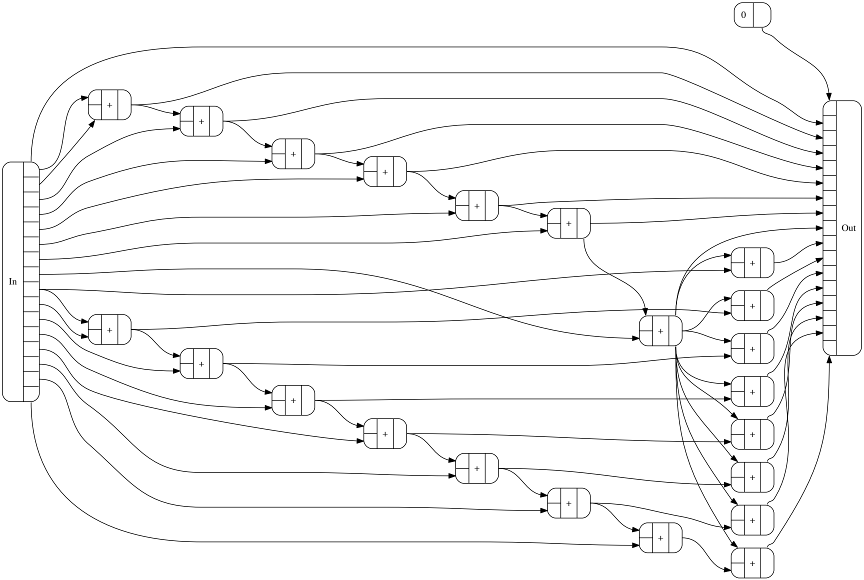
Pair ∘ *LVec* 8

work: 22, depth: 8



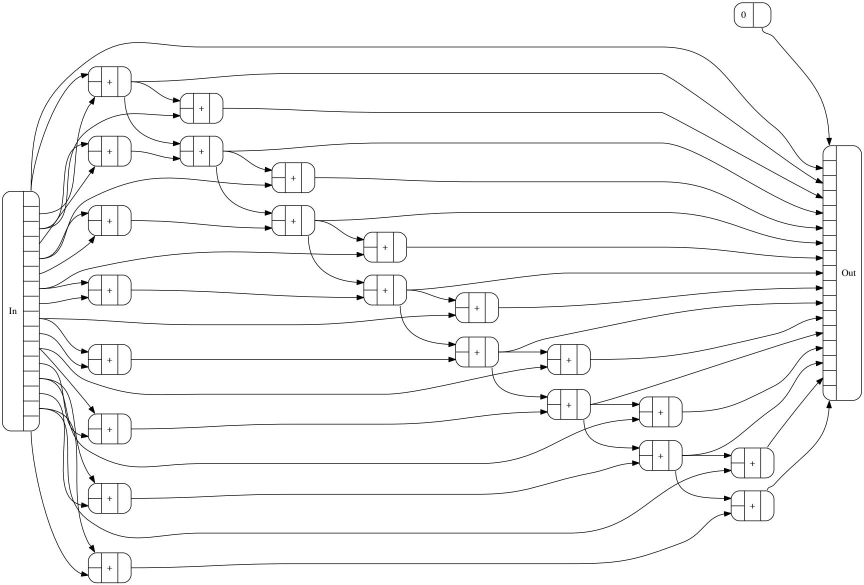
2×8

work: 22, depth: 8



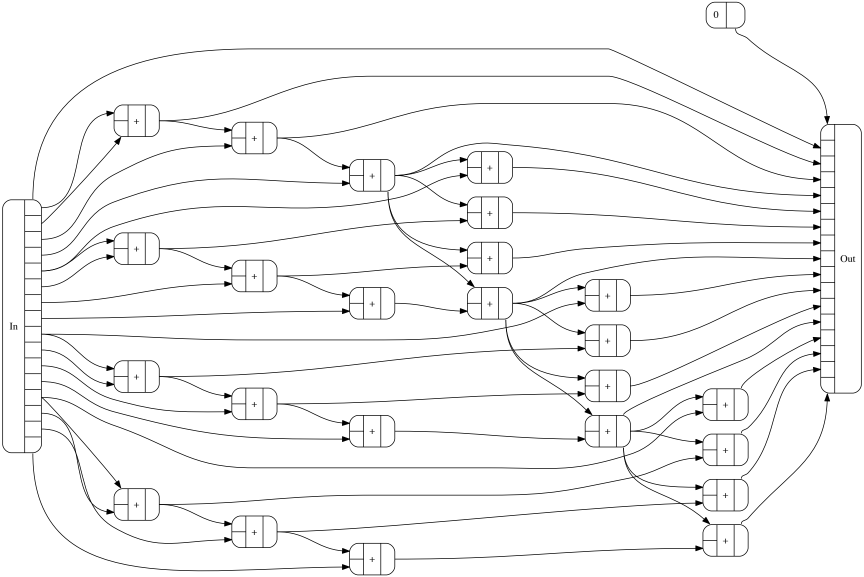
8×2

work: 22, depth: 8



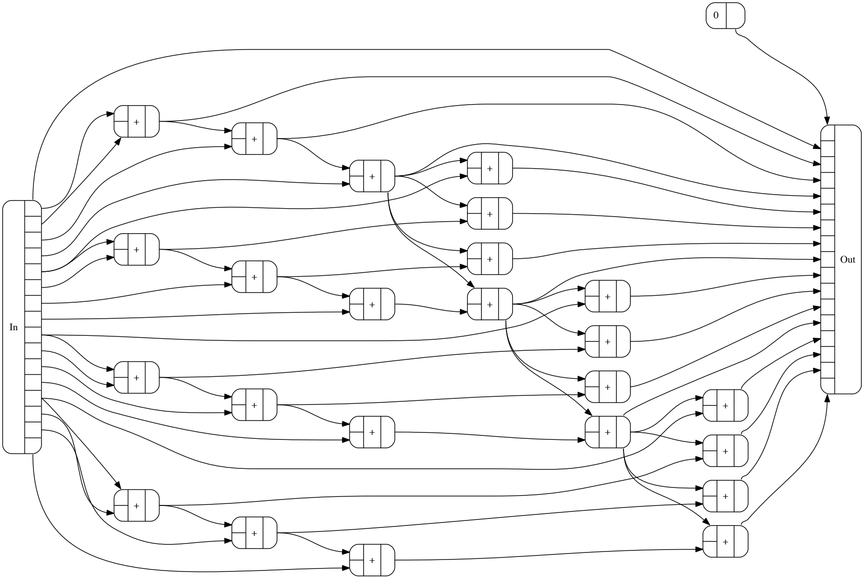
4×4

work: 24, depth: 6



4^2

work: 24, depth: 6



Functor exponentiation as type families

$$f^n = \overbrace{f \circ \cdots \circ f}^{n \text{ times}}$$

Right-associated/top-down:

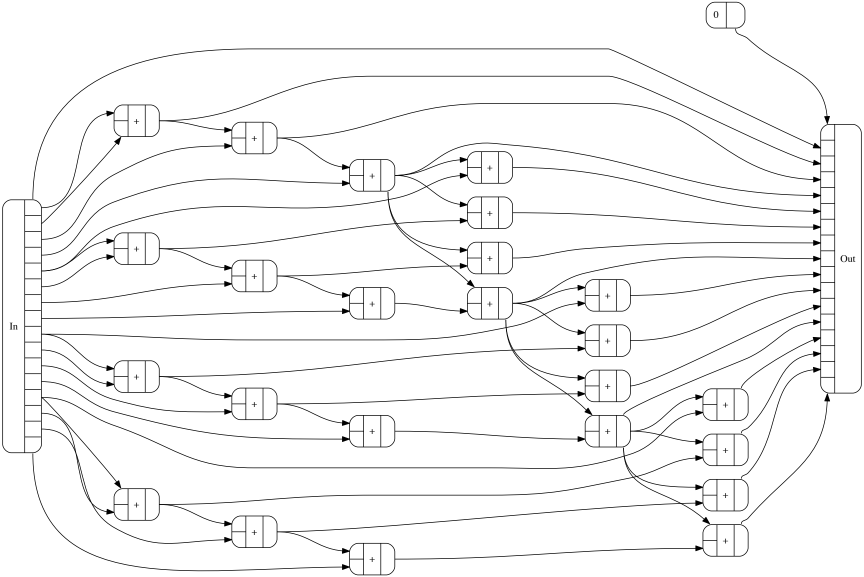
$$\begin{array}{l}\textbf{type family } \mathit{RPow}\ h\ n \textbf{ where}\\ \quad \mathit{RPow}\ h\ Z \quad\quad\ = \mathit{Par}_1\\ \quad \mathit{RPow}\ h\ (S\ n) = h \circ \mathit{RPow}\ h\ n\end{array}$$

Left-associated/bottom-up:

$$\begin{array}{l}\textbf{type family } \mathit{LPow}\ h\ n \textbf{ where}\\ \quad \mathit{LPow}\ h\ Z \quad\quad\ = \mathit{Par}_1\\ \quad \mathit{LPow}\ h\ (S\ n) = \mathit{LPow}\ h\ n \circ h\end{array}$$

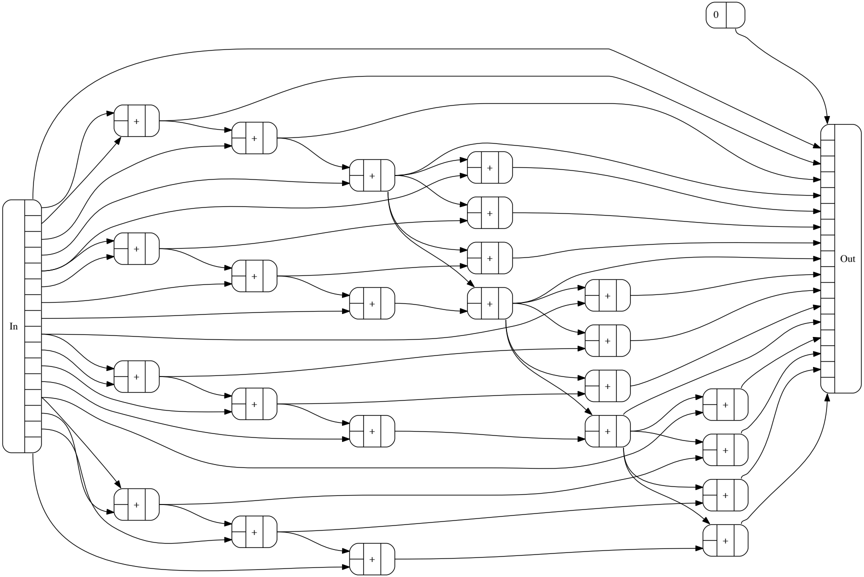
RPow (*LVec* 4) 2

work: 24, depth: 6

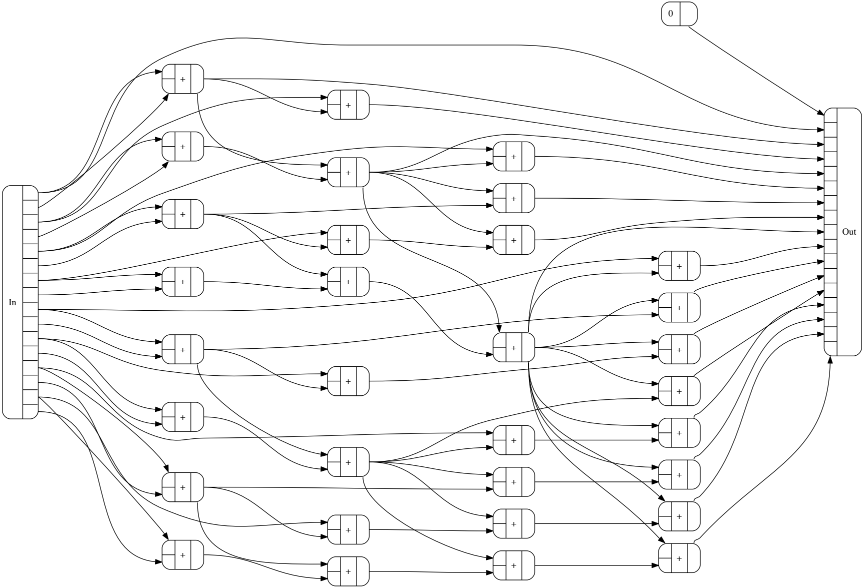


4^2

work: 24, depth: 6

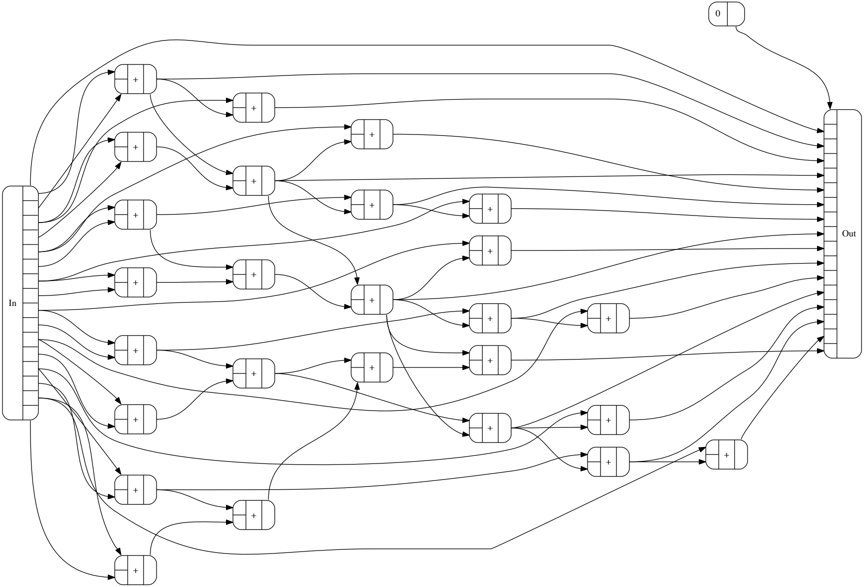


$\overrightarrow{2^4} = 2 \times (2 \times (2 \times 2))$ work: 32, depth: 4



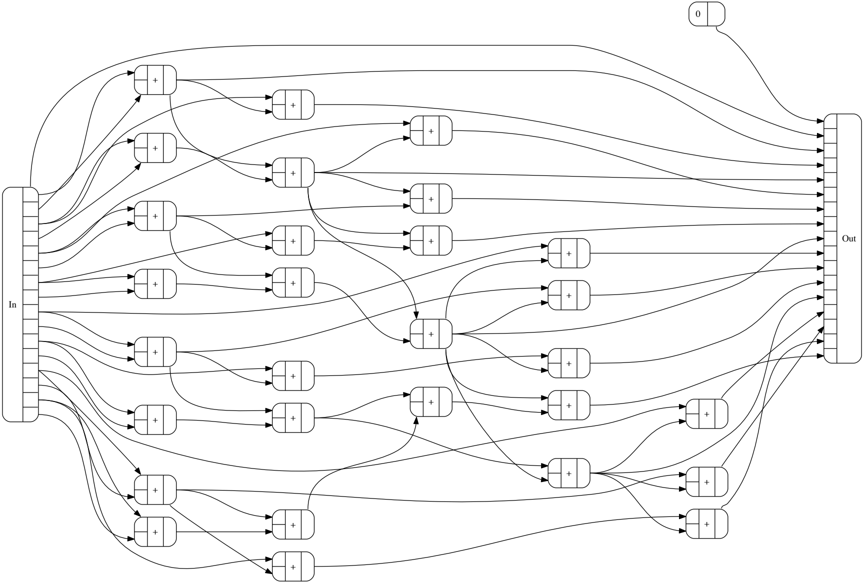
$\overleftarrow{2^4} = ((2 \times 2) \times 2) \times 2$

work: 26, depth: 6



$2^4 = (2^2)^2 = (2 \times 2) \times (2 \times 2)$

work: 29, depth: 5



Bushes

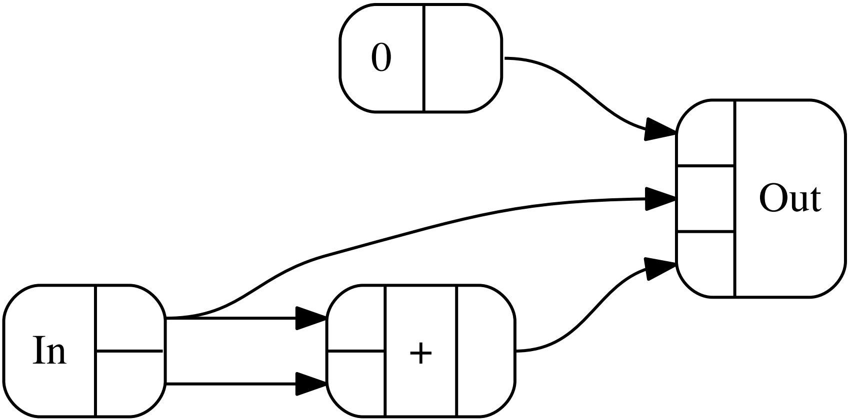
type family *Bush n* **where**
  $Bush\ Z \quad\ \ = Pair$
  $Bush\ (S\ n) = Bush\ n \circ Bush\ n$

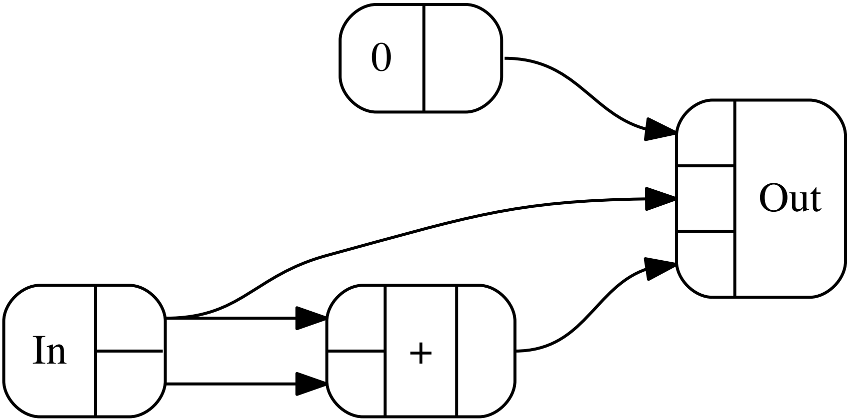
Notes:

- Composition-balanced counterpart to *LPow* and *RPow*.
- Variation of *Bush* type in *Nested Datatypes* by Bird & Meertens.
- Size 2^{2^n}, i.e., $2, 4, 16, 256, 65536, \ldots$.
- Easily generalizes beyond pairing and squaring.

Bush 0

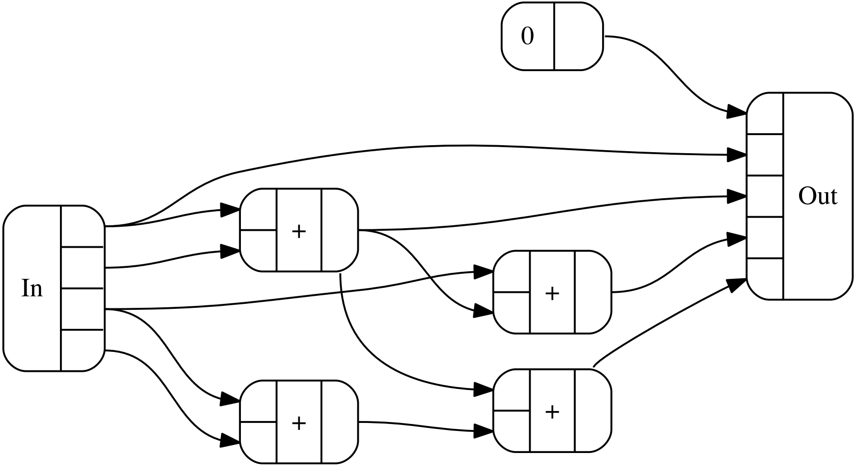
work: 1, depth: 1




0
Out
In
+

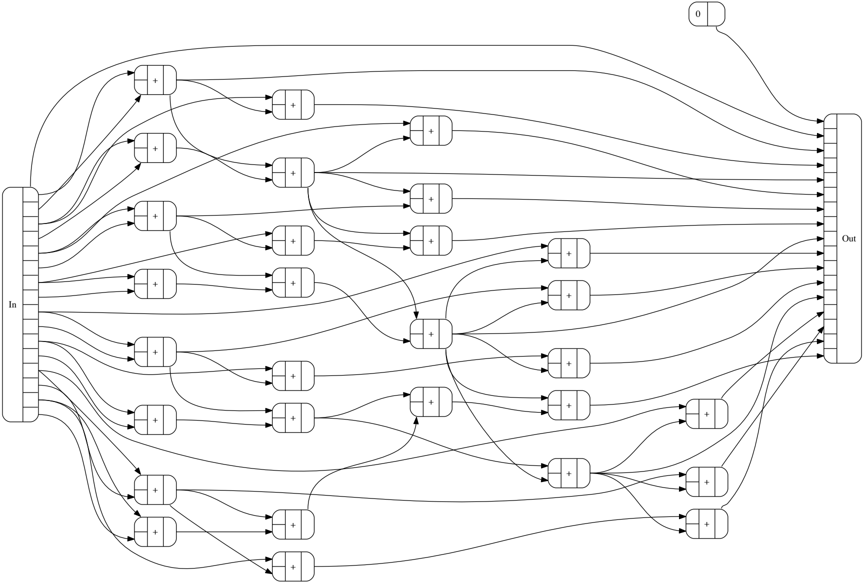
2^{2^1}

work: 4, depth: 2



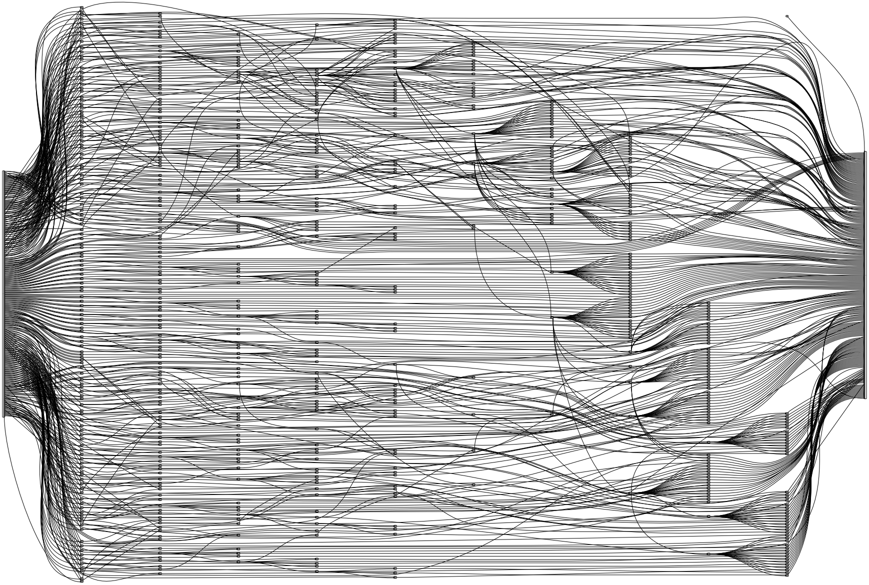
2^{2^2}

work: 29, depth: 5



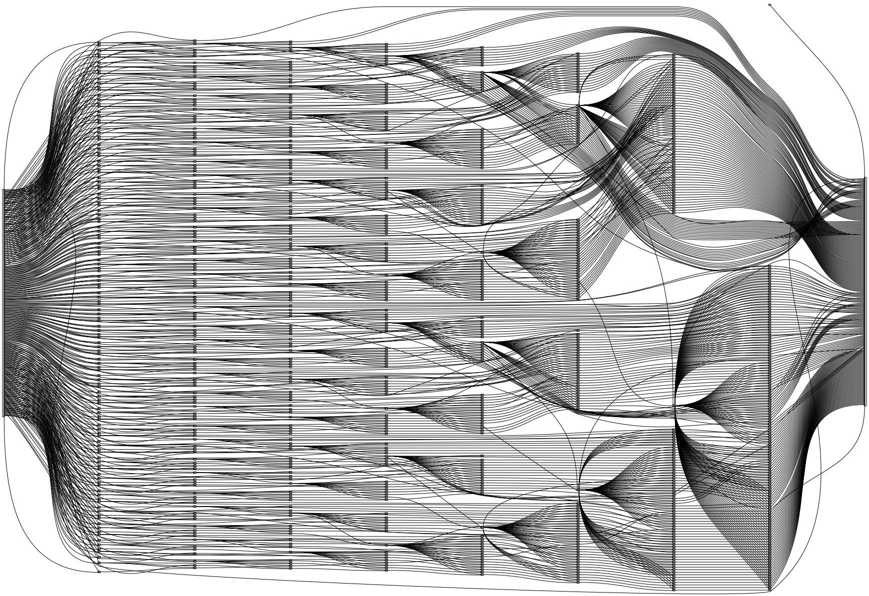
2^{2^3}

work: 718, depth: 10

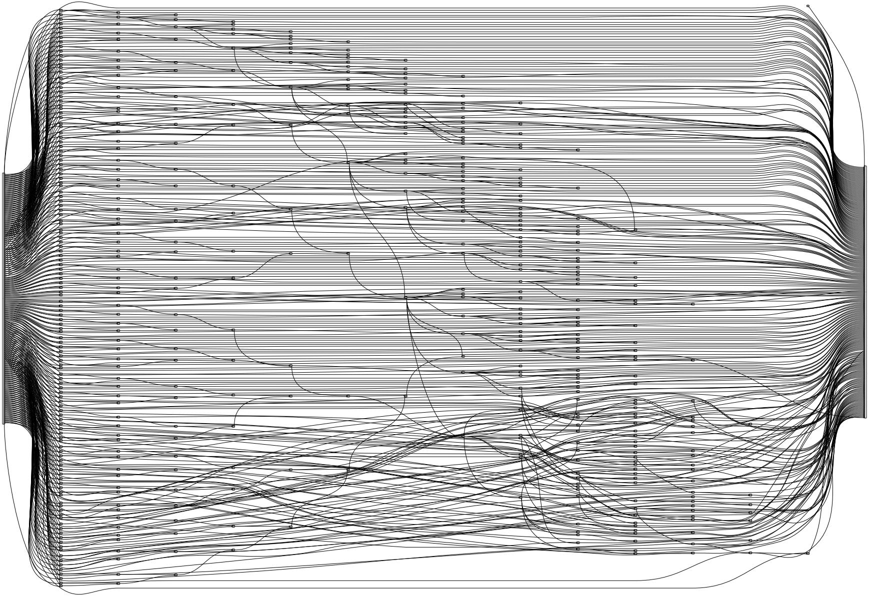


$\overrightarrow{2^8}$

work: 1024, depth: 8



work: 502, depth: 14



Generic parallel scan

- Parallel scan: useful for many parallel algorithms.
- Parallel programming without arrays:
  - Safety (no indexing errors).
  - Functor shape guides algorithm shape.
- Generic programming:
  - Define per functor building block.
  - Use directly, *or*
  - automatically via (perhaps derived) encodings.
  - Infinite variations, easily explored and guaranteed correct.
- Related talk: Generic FFT
- Paper: Generic parallel functional programming

Extras

- Data encodings
- Convenient packaging
- Application: polynomial evaluation
- Application: parallel addition

Data encodings

From *GHC.Generics*:

$$
\begin{array}{l}
\textbf{class}\ \mathit{Generic}_1\ f\ \textbf{where} \\
\quad \textbf{type}\ \mathit{Rep}_1\ f :: * \to * \\
\quad \mathit{from}_1 :: f\ a \to \mathit{Rep}_1\ f\ a \\
\quad \mathit{to}_1 \quad :: \mathit{Rep}_1\ f\ a \to f\ a
\end{array}
$$

For regular algebraic data types, say "... **deriving** $\mathit{Generic}_1$".

Scan class

$$
\begin{array}{l}
\textbf{class } \mathit{Functor}\ f \Rightarrow \mathit{LScan}\ f \textbf{ where} \\
\quad \mathit{lscan} :: \mathit{Monoid}\ a \Rightarrow f\ a \to f\ a \times a \\
\\
\quad \textbf{default } \mathit{lscan} :: (\mathit{Generic}_1\ f, \mathit{LScan}\ (\mathit{Rep}_1\ f)) \\
\qquad\qquad\qquad\qquad \Rightarrow \mathit{Monoid}\ a \Rightarrow f\ a \to f\ a \times a \\
\quad \mathit{lscan} = \mathit{first}\ \mathit{to}_1 \circ \mathit{lscan} \circ \mathit{from}_1
\end{array}
$$

Vector type families

Right-associated:

$$\textbf{type family } RVec\ n \textbf{ where}$$
$$RVec\ Z = U_1$$
$$RVec\ (S\ n) = Par_1 \times RVec\ n$$

Left-associated:

$$\textbf{type family } LVec\ n \textbf{ where}$$
$$LVec\ Z = U_1$$
$$LVec\ (S\ n) = LVec\ n \times Par_1$$

Vector GADTs

```
data RVec :: Nat → * → * where
  ZVec ::                  RVec Z     a
  (:<) :: a → RVec n a → RVec (S n) a

instance Generic1 (RVec Z) where
  type Rep1 (RVec Z) = U1
  from1 ZVec = U1
  to1 U1 = ZVec

instance Generic1 (RVec (S n)) where
  type Rep1 (RVec (S n)) = Par1 × RVec n
  from1 (a :< as) = Par1 a × as
  to1 (Par1 a × as) = a :< as

instance                   LScan (RVec Z)
instance LScan (RVec n) ⇒ LScan (RVec (S n))
```

Plus *Functor*, *Applicative*, *Foldable*, *Traversable*, *Monoid*, *Key*, ...

Functor exponentiation type families

$$f^n = \overbrace{f \circ \cdots \circ f}^{n \text{ times}}$$

Right-associated/top-down:

$$\begin{array}{l} \textbf{type family } \mathit{RPow}\ h\ n\ \textbf{where} \\ \quad \mathit{RPow}\ h\ Z \quad\quad\ = \mathit{Par}_1 \\ \quad \mathit{RPow}\ h\ (S\ n) = h \circ \mathit{RPow}\ h\ n \end{array}$$

Left-associated/bottom-up:

$$\begin{array}{l} \textbf{type family } \mathit{LPow}\ h\ n\ \textbf{where} \\ \quad \mathit{LPow}\ h\ Z \quad\quad\ = \mathit{Par}_1 \\ \quad \mathit{LPow}\ h\ (S\ n) = \mathit{LPow}\ h\ n \circ h \end{array}$$

Functor exponentiation GADTs

$$f^n = \overbrace{f \circ \cdots \circ f}^{n \text{ times}}$$

Right-associated/top-down:

$$\begin{array}{l}
\textbf{data}\ \mathit{RPow} :: (* \to *) \to \mathit{Nat} \to * \to *\ \textbf{where} \\
\quad L :: a \to \mathit{RPow}\ h\ Z\ a \\
\quad B :: h\ (\mathit{RPow}\ h\ n\ a) \to \mathit{RPow}\ h\ (S\ n)\ a
\end{array}$$

Left-associated/bottom-up:

$$\begin{array}{l}
\textbf{data}\ \mathit{LPow} :: (* \to *) \to \mathit{Nat} \to * \to *\ \textbf{where} \\
\quad L :: a \to \mathit{LPow}\ h\ Z\ a \\
\quad B :: \mathit{LPow}\ h\ n\ (h\ a) \to \mathit{LPow}\ h\ (S\ n)\ a
\end{array}$$

Plus $\mathit{Generic}_1$, $\mathit{Functor}$, $\mathit{Foldable}$, $\mathit{Traversable}$, Monoid, Key, ...

Some convenient packaging

$$\begin{aligned}
&lscanAla :: \forall n\ o\ f.(Newtype\ n, o \sim O\ n, LScan\ f, Monoid\ n) \\
&\qquad \Rightarrow f\ o \rightarrow f\ o \times o \\
&lscanAla = (fmap\ unpack \times unpack) \circ lscan \circ fmap\ (pack\ @n)
\end{aligned}$$

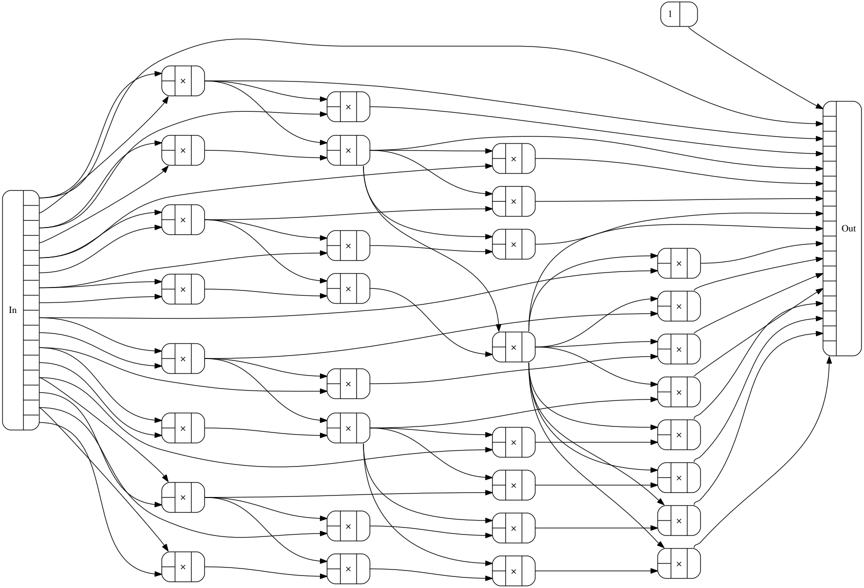
$$\begin{aligned}
lsums &= lscanAla\ @(Sum\ a) \\
lproducts &= lscanAla\ @(Product\ a) \\
lalls &= lscanAla\ @All \\
&\ldots
\end{aligned}$$

Some simple uses:

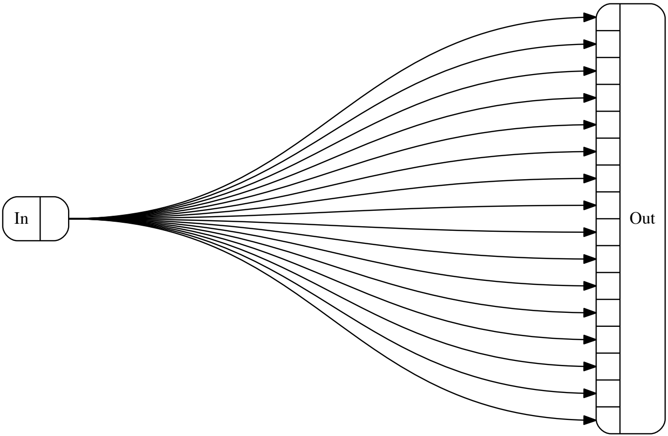
$$\begin{aligned}
multiples &= lsums \circ point \\
powers &= lproducts \circ point
\end{aligned}$$

lproducts @$\overrightarrow{2^4}$

work: 32, depth: 4

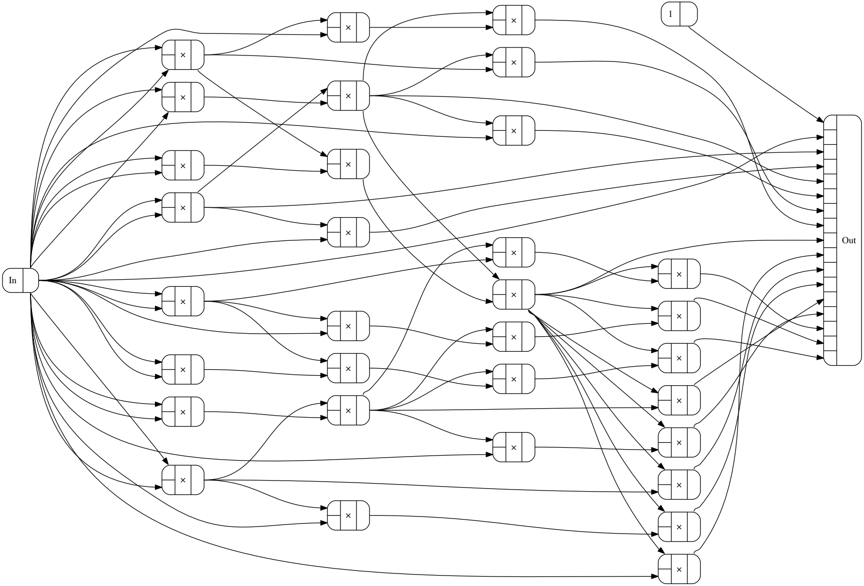


$point\ @\overrightarrow{2^4}$



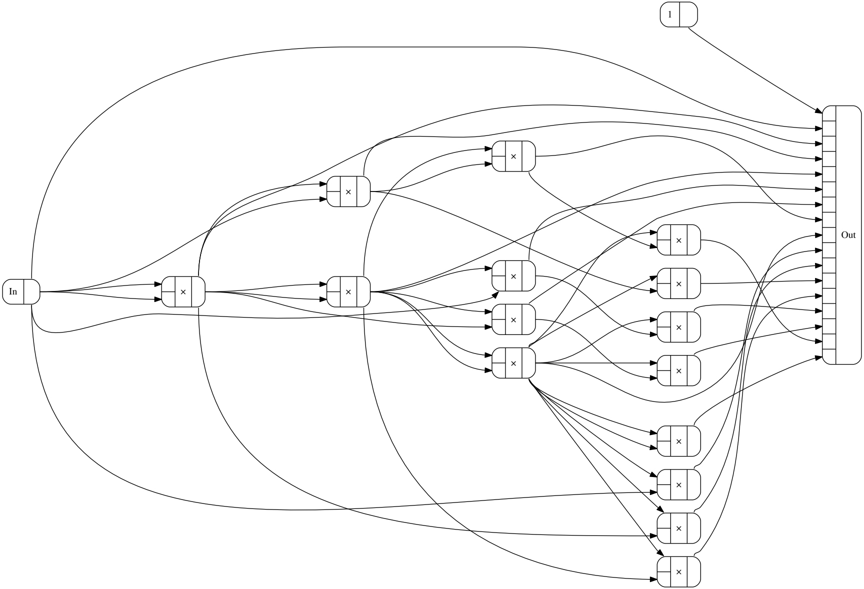
powers $@\overrightarrow{2^4}$

work: 32, depth: 4



$\mathit{powers} \mathbin{@} \overrightarrow{2^4}$ — with CSE

work: 15, depth: 4



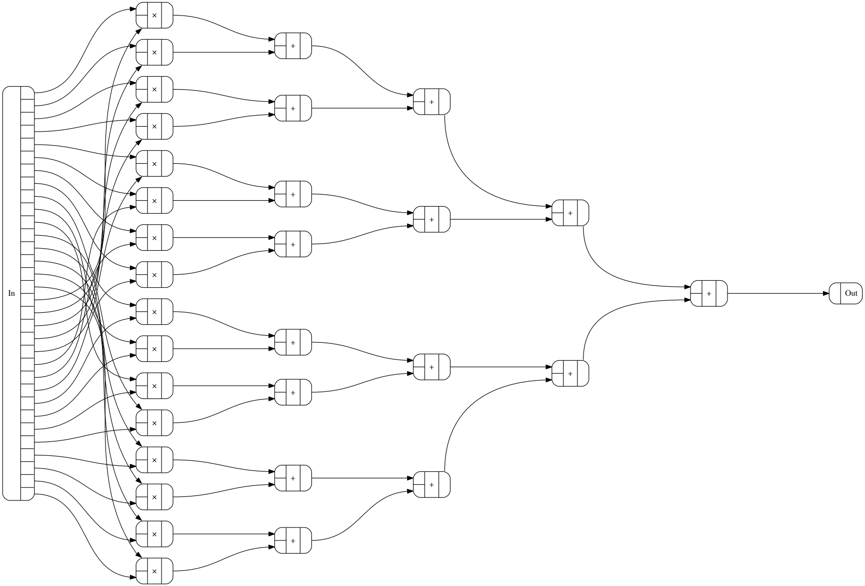
Example: polynomial evaluation

$$\begin{aligned}
&\mathit{evalPoly} :: (\mathit{LScan}\ f, \mathit{Foldable}\ f, \mathit{Zip}\ f, \mathit{Pointed}\ f, \mathit{Num}\ a) \\
&\qquad\qquad \Rightarrow f\ a \rightarrow a \rightarrow a \\
&\mathit{evalPoly}\ \mathit{coeffs}\ x = \mathit{coeffs} \cdot \mathit{fst}\ (\mathit{powers}\ x)
\end{aligned}$$

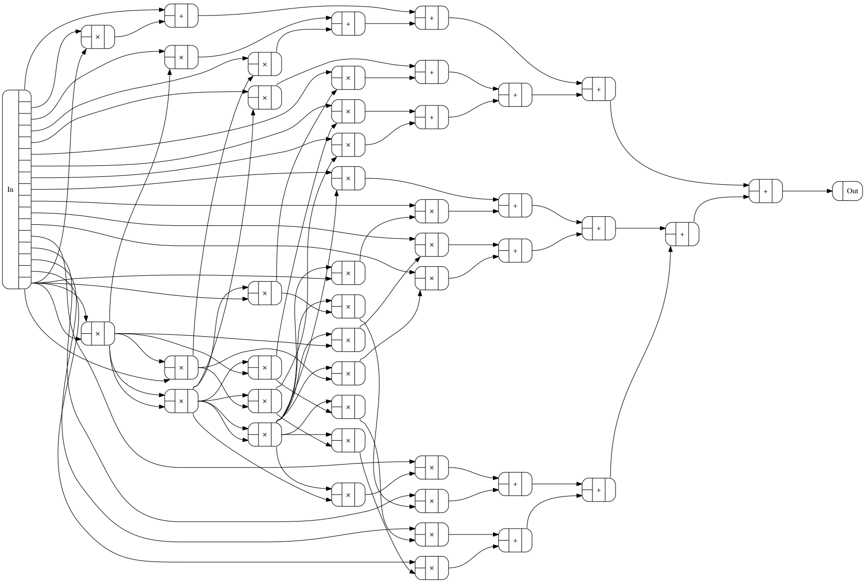
$$\begin{aligned}
&(\cdot) :: (\mathit{Foldable}\ f, \mathit{Zip}\ f, \mathit{Num}\ a) \Rightarrow f\ a \rightarrow f\ a \rightarrow a \\
&u \cdot v = \mathit{sum}\ (\mathit{zipWith}\ (\times)\ u\ v)
\end{aligned}$$

$(\cdot)\ @\overrightarrow{2^4}$

work: 16+15, depth: 5



$evalPoly\ @\overrightarrow{2^4}$ work: 29+15, depth: 9



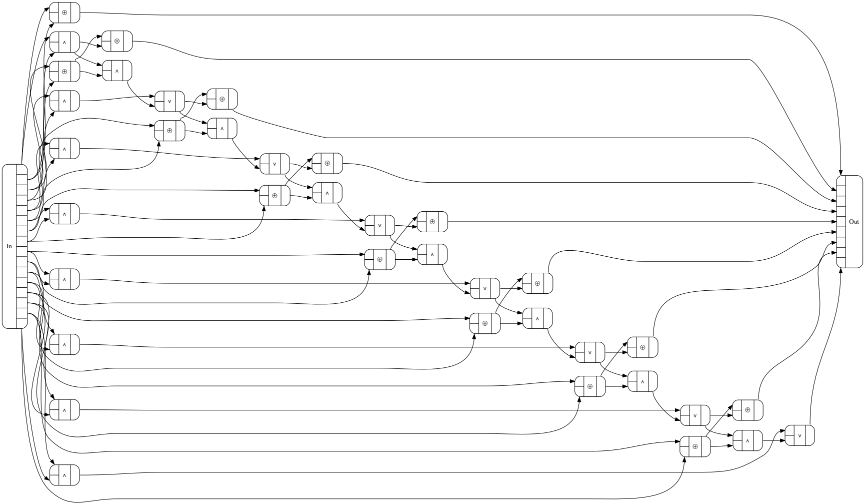
Addition

Generate and propagate carries:

$$\textbf{data}\ \mathit{PropGen} = \mathit{PropGen}\ \mathit{Bool}\ \mathit{Bool}$$

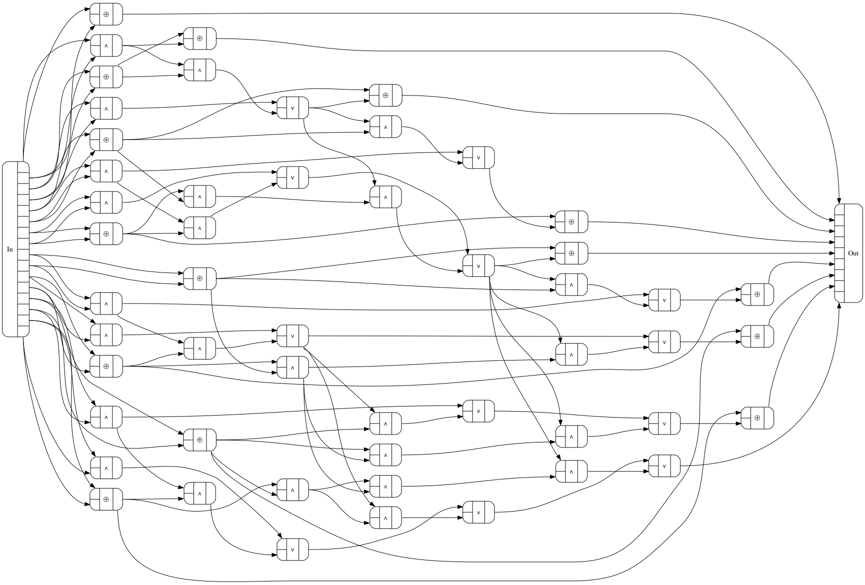
$$\begin{aligned}
&\mathit{propGen} :: \mathit{Bool} \rightarrow \mathit{Bool} \rightarrow \mathit{PropGen} \\
&\mathit{propGen}\ a\ b = \mathit{PropGen}\ (a \oplus b)\ (a \wedge b) \quad \text{-- half adder}
\end{aligned}$$

$$\begin{aligned}
&\textbf{instance}\ \mathit{Monoid}\ \mathit{PropGen}\ \textbf{where} \\
&\quad \varepsilon = \mathit{PropGen}\ \mathit{True}\ \mathit{False} \\
&\quad \mathit{PropGen}\ \mathit{pa}\ \mathit{ga} \diamond \mathit{PropGen}\ \mathit{pb}\ \mathit{gb} = \\
&\qquad \mathit{PropGen}\ (\mathit{pa} \wedge \mathit{pb})\ ((\mathit{ga} \wedge \mathit{pb}) \vee \mathit{gb})
\end{aligned}$$


In
Out

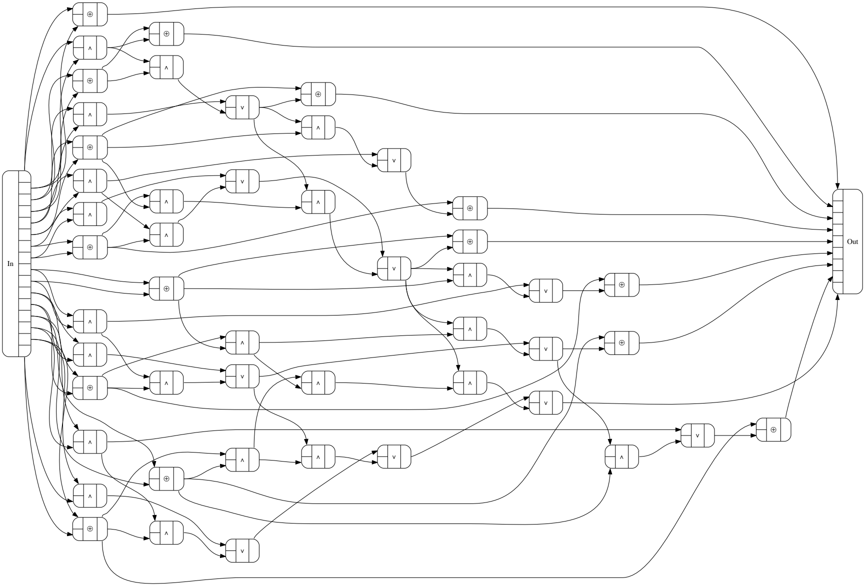
scanAdd @$\overrightarrow{2}^3$

work: 52, depth: 8



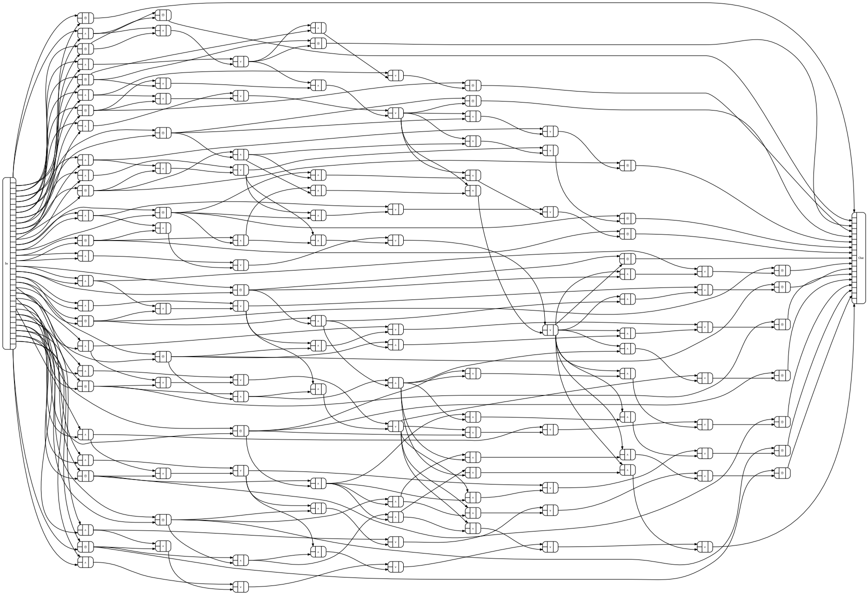
$scanAdd\ @\overleftarrow{2^3}$

work: 49, depth: 10

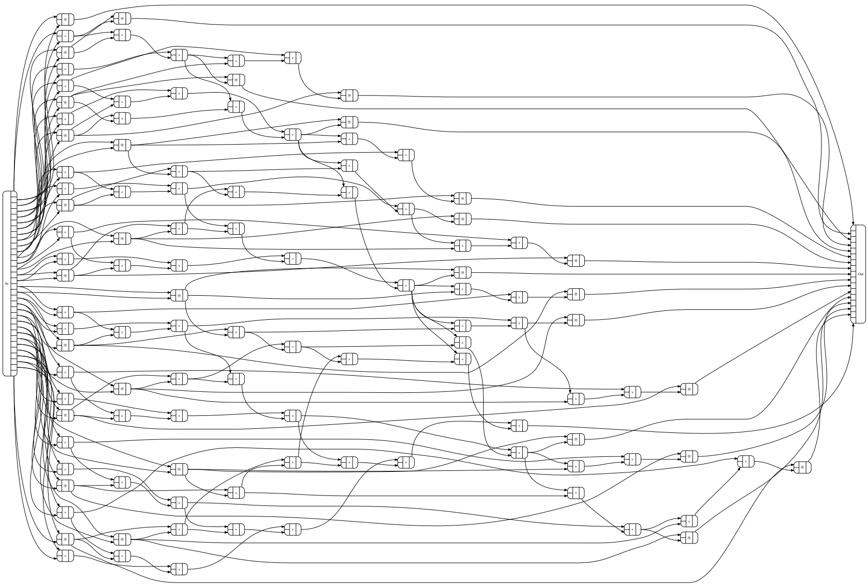


scanAdd @$\overrightarrow{2^4}$

work: 128, depth: 10



$scanAdd\ @\overleftarrow{2^4}$ work: 110, depth: 14



scanAdd @2^{2^2} work: 119, depth: 12