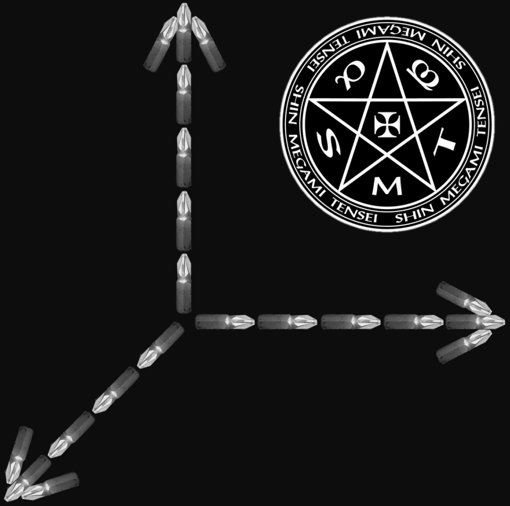
SMT: Bit Vectors

Florian Märkl
June 25, 2020



Motivation

```
uint8_t a = 200;
uint8_t b = a + 98;
assert(b > a);
```

Motivation

```
uint8_t a = 200;
uint8_t b = a + 98;
assert(b > a);
```

$$a + 98 > a$$

Motivation

```
uint8_t a = 200;
uint8_t b = a + 98;
assert(b > a);
```

$a + 98 > a$ ✓

Motivation

```
uint8_t a = 200;
uint8_t b = a + 98;
assert(b > a);
```

$a = 200$

$a + 98 > a$ ✓

Motivation

```
uint8_t a = 200;
uint8_t b = a + 98;
assert(b > a);
```

$a = 200$

$b = 42$

$a + 98 > a$ ✓

Motivation

```c
uint8_t a = 200;
uint8_t b = a + 98;
assert(b > a);
```

$a = 200$

$b = 42$

$\lightning$

$a + 98 > a$ ✓

Motivation

```
uint8_t a = 200;
uint8_t b = a + 98;
assert(b > a);
```

$a = 200$

$b = 42$

$\lightning$

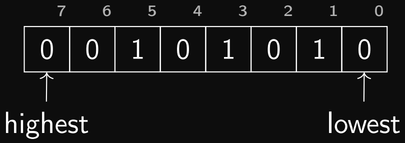
Bit Vectors

0	0	1	0	1	0	1	0

Bit Vectors

7	6	5	4	3	2	1	0
0	0	1	0	1	0	1	0

Bit Vectors



Bit Vectors

7	6	5	4	3	2	1	0
0	0	1	0	1	0	1	0

highest   lowest

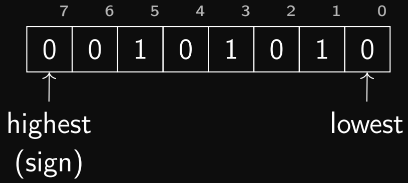
Unsigned: **Base-2**

Bit Vectors

7	6	5	4	3	2	1	0
0	0	1	0	1	0	1	0
highest							lowest

Unsigned: **Base-2**
Signed: **Two's complement**

Bit Vectors



Unsigned: **Base-2**

Signed: **Two's complement**

Syntax

$$
\begin{array}{rcl}
\mathit{term} & \rightarrow & \mathit{term}\ \mathit{op}\ \mathit{term} \mid \mathit{var\text{—}identifier} \mid {\sim}\mathit{term} \mid \mathit{constant} \\
& & \mid \mathit{atom}\ ?\ \mathit{term} : \mathit{term} \mid \mathit{term}[\mathit{constant} : \mathit{constant}] \\
& & \mid \mathrm{ext}\,(\mathit{term}) \\
\mathit{op} & \rightarrow & + \mid - \mid \cdot \mid / \mid \ll \mid \gg \mid \& \mid \;|\; \mid \oplus \mid \circ
\end{array}
$$

Syntax

$$
\begin{aligned}
term &\rightarrow term \;\; op \;\; term \mid var\text{—}identifier \mid {\sim}term \mid constant \\
&\quad \mid atom \;?\; term : term \mid term[\,constant : constant\,] \\
&\quad \mid \mathrm{ext}\,(\,term\,) \\
op &\rightarrow + \mid - \mid \cdot \mid / \mid \ll \mid \gg \mid \& \mid | \mid \oplus \mid \circ
\end{aligned}
$$

$$
atom \rightarrow term < term \mid term = term \mid term[\,constant\,]
$$

Syntax

$$
\begin{array}{rcl}
term & \rightarrow & term \;\; op \;\; term \;\mid\; var\text{—}identifier \;\mid\; \sim term \;\mid\; constant \\
 & \mid & atom \;?\; term \;:\; term \;\mid\; term[\,constant : constant\,] \\
 & \mid & \mathrm{ext}\,(\,term\,) \\
op & \rightarrow & + \mid - \mid \cdot \mid / \mid \ll \mid \gg \mid \& \mid \;|\; \mid \oplus \mid \circ
\end{array}
$$

$$
atom \;\rightarrow\; term < term \;\mid\; term = term \;\mid\; term[\,constant\,]
$$

$$
formula \;\rightarrow\; formula \wedge formula \;\mid\; \neg formula \;\mid\; (\,formula\,) \;\mid\; atom
$$

Syntax

$$
\begin{aligned}
term &\rightarrow term\ \ op\ \ term \mid var{-}identifier \mid {\sim}term \mid constant \\
&\mid atom\ ?\ term : term \mid term[\,constant : constant\,] \\
&\mid \text{ext}\,(\,term\,) \\
op &\rightarrow + \mid - \mid \cdot \mid / \mid \ll \mid \gg \mid \& \mid \,|\, \mid \oplus \mid \circ
\end{aligned}
$$

Bit Vector: **size + signed|unsigned**

$$
atom \rightarrow term < term \mid term = term \mid term[\,constant\,]
$$

$$
formula \rightarrow formula \wedge formula \mid \neg formula \mid (\,formula\,) \mid atom
$$

Syntax

$$\begin{array}{rcl} term & \rightarrow & term \;\; op \;\; term \mid var\text{–}identifier \mid {\sim}term \mid constant \\ & & \mid \; atom \; ? \; term \; : \; term \mid term[\,constant : constant\,] \\ & & \mid \; \text{ext}\,(\,term\,) \\ op & \rightarrow & + \mid - \mid \cdot \mid / \mid \ll \mid \gg \mid \& \mid \; \mid \; \mid \oplus \mid \circ \end{array}$$

Bit Vector: **size + signed|unsigned**

$$atom \;\rightarrow\; term < term \mid term = term \mid term[\,constant\,]$$

Boolean

$$formula \;\rightarrow\; formula \wedge formula \mid \neg formula \mid (\,formula\,) \mid atom$$

Syntax

$$
\begin{array}{rcl}
\mathit{term} & \rightarrow & \mathit{term}\ \mathit{op}\ \mathit{term} \mid \mathit{var{-}identifier} \mid \sim\!\mathit{term} \mid \mathit{constant} \\
& & \mid\ \mathit{atom}\ ?\ \mathit{term} : \mathit{term} \mid \mathit{term}[\mathit{constant} : \mathit{constant}] \\
& & \mid\ \mathrm{ext}(\mathit{term}) \\
\mathit{op} & \rightarrow & + \mid - \mid \cdot \mid / \mid \ll \mid \gg \mid \& \mid | \mid \oplus \mid \circ
\end{array}
$$

Bit Vector: **size + signed|unsigned**

$$
\mathit{atom} \rightarrow \mathit{term} < \mathit{term} \mid \mathit{term} = \mathit{term} \mid \mathit{term}[\mathit{constant}]
$$

Boolean

$$
\mathit{formula} \rightarrow \mathit{formula} \wedge \mathit{formula} \mid \neg\mathit{formula} \mid (\mathit{formula}) \mid \mathit{atom}
$$

Boolean

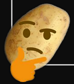
Solver: Implementation

https://github.com/thestr4ng3r/shida

Solver: Flattening

Bit Vector Formula

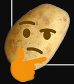
$$10 = 11 \oplus a$$



Solver: Flattening

Bit Vector Formula

$10 = 11 \oplus a$



$\xrightarrow{\text{Flattening}}$

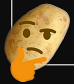
Propositional Formula

False = True $\oplus$ "a[0]"
True = True $\oplus$ "a[1]"

Solver: Flattening

Bit Vector Formula

$10 = 11 \oplus a$



Flattening

Propositional Formula

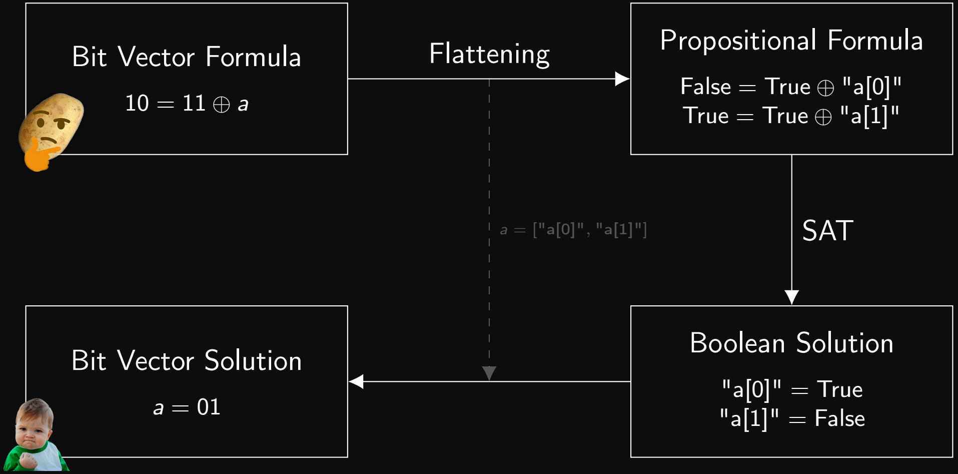
False $=$ True $\oplus$ "a[0]"
True $=$ True $\oplus$ "a[1]"

SAT

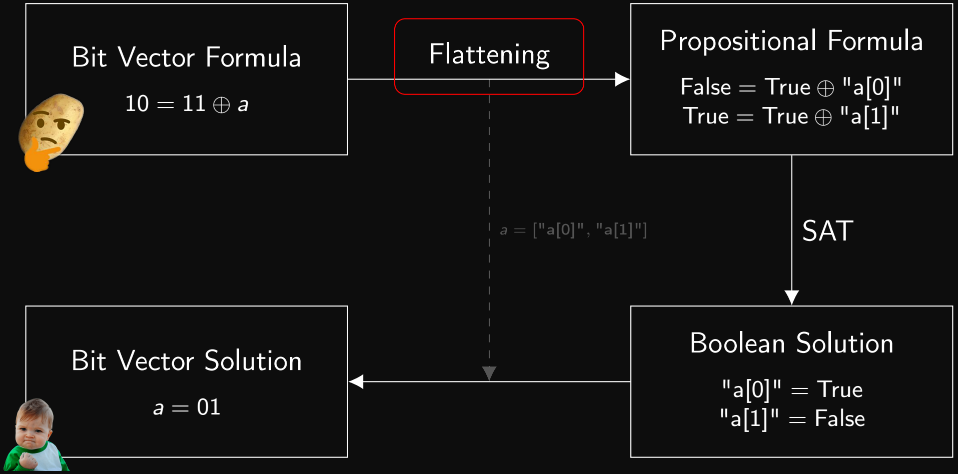
Boolean Solution

"a[0]" $=$ True
"a[1]" $=$ False

Solver: Flattening



Solver: Flattening



Flattening: High-Level Algorithm

Example:
$a < 011 \land \neg a < 010$

$\boxed{011}$ $\boxed{a < 011}$ $\boxed{a}$ $\boxed{a < 010}$ $\boxed{010}$

```
flatten :: Formula -> Propositional.Formula
flatten f =
```

Flattening: High-Level Algorithm

Example:

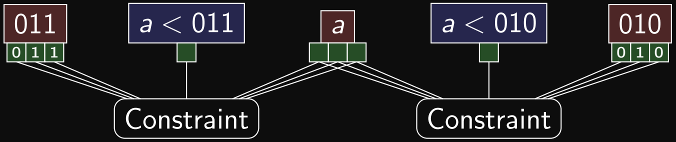
$a < 011 \wedge \neg a < 010$


```
flatten :: Formula -> Propositional.Formula
flatten f =
  let
    termProps        = reserveVarsForAll (terms f)
    atomProps        = reserveVarsForAll (atoms f)
```

Flattening: High-Level Algorithm

Example:

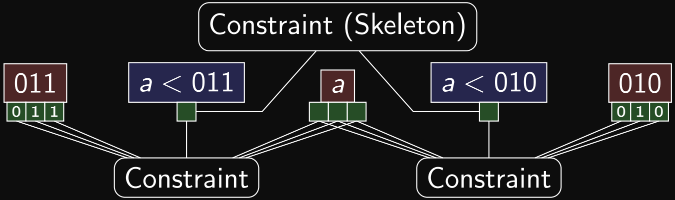
$a < 011 \wedge \neg a < 010$




```
flatten :: Formula -> Propositional.Formula
flatten f =
  let
    termProps        = reserveVarsForAll (terms f)
    atomProps        = reserveVarsForAll (atoms f)
    termConstraints = {termConstraint atomProps termProps term | term ∈ (terms f)}
    atomConstraints = {atomConstraint atomProps termProps atom | atom ∈ (atoms f)}
```

Flattening: High-Level Algorithm

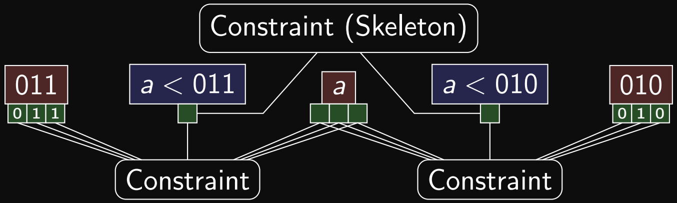
Example:
$a < 011 \wedge \neg a < 010$




```
flatten :: Formula -> Propositional.Formula
flatten f =
  let
    termProps        = reserveVarsForAll (terms f)
    atomProps        = reserveVarsForAll (atoms f)
    termConstraints  = {termConstraint atomProps termProps term | term ∈ (terms f)}
    atomConstraints  = {atomConstraint atomProps termProps atom | atom ∈ (atoms f)}
    skel             = skeleton atomProps f
```

Flattening: High-Level Algorithm

Example:
$a < 011 \land \neg a < 010$




```
flatten :: Formula -> Propositional.Formula
flatten f =
  let
    termProps        = reserveVarsForAll (terms f)
    atomProps        = reserveVarsForAll (atoms f)
    termConstraints = {termConstraint atomProps termProps term | term ∈ (terms f)}
    atomConstraints = {atomConstraint atomProps termProps atom | atom ∈ (atoms f)}
    skel             = skeleton atomProps f
  in (skel ∪ termConstraints ∪ atomConstraints)
```

Flattening: Skeleton

$$a + b < 101010 \land \neg(b = 010101)$$

Flattening: Skeleton

$$a + b < 101010 \land \neg(b = 010101)$$

$$\downarrow$$

atomProps["a + b < 101010"] $\land$ $\neg$atomProps["b = 010101"]

Flattening: Bitwise Operators and Equality

$l \oplus r$

Flattening: Bitwise Operators and Equality

$$l \oplus r$$

$$\downarrow$$

$$(i \oplus r)[0] \iff (l[0] \oplus r[0])$$
$$(i \oplus r)[1] \iff (l[1] \oplus r[1])$$
$$(i \oplus r)[2] \iff (l[2] \oplus r[2])$$
$$\dots$$

Flattening: Bitwise Operators and Equality

$$l \oplus r$$

$$\downarrow$$

$$(i \oplus r)[0] \iff (l[0] \oplus r[0])$$
$$(i \oplus r)[1] \iff (l[1] \oplus r[1])$$
$$(i \oplus r)[2] \iff (l[2] \oplus r[2])$$
$$\dots$$

$$(l \oplus r)[i] \iff (l[i] \oplus r[i])$$

Flattening: Bitwise Operators and Equality

$$l \oplus r$$

$$\downarrow$$

$$(i \oplus r)[0] \iff (l[0] \oplus r[0])$$
$$(i \oplus r)[1] \iff (l[1] \oplus r[1])$$
$$(i \oplus r)[2] \iff (l[2] \oplus r[2])$$
$$\dots$$

$$(l \oplus r)[i] \iff (l[i] \oplus r[i])$$

$$l = r$$

Flattening: Bitwise Operators and Equality

$$l \oplus r$$

$$\downarrow$$

$$(i \oplus r)[0] \iff (l[0] \oplus r[0])$$
$$(i \oplus r)[1] \iff (l[1] \oplus r[1])$$
$$(i \oplus r)[2] \iff (l[2] \oplus r[2])$$
$$\ldots$$

$$(l \oplus r)[i] \iff (l[i] \oplus r[i])$$

$$l = r$$

$$\downarrow$$

$$(l = r) \iff$$

Flattening: Bitwise Operators and Equality

$$l \oplus r$$

$$\downarrow$$

$$(i \oplus r)[0] \iff (l[0] \oplus r[0])$$
$$(i \oplus r)[1] \iff (l[1] \oplus r[1])$$
$$(i \oplus r)[2] \iff (l[2] \oplus r[2])$$
$$\dots$$

$$(l \oplus r)[i] \iff (l[i] \oplus r[i])$$

$$l = r$$

$$\downarrow$$

$$(l = r) \iff \bigwedge_i (l[i] \iff r[i])$$

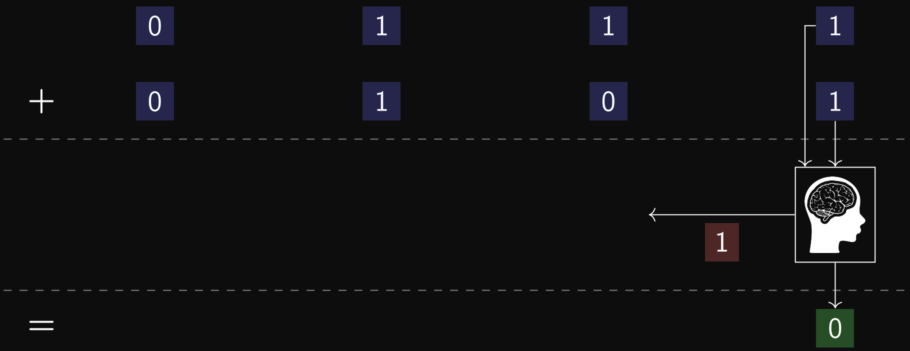
Flattening: Addition

$$
\begin{array}{ccccc}
 & 0 & 1 & 1 & 1 \\
+ & 0 & 1 & 0 & 1 \\
\hline
\\
\hline
= & & & &
\end{array}
$$

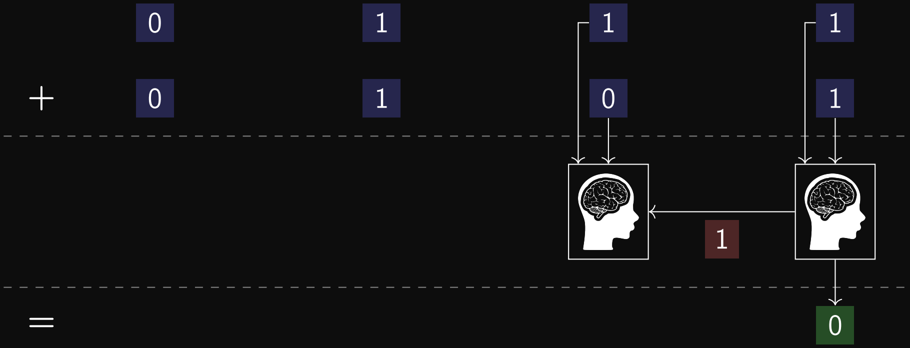
Flattening: Addition

	0	1	1	1
+	0	1	0	1
				
=				

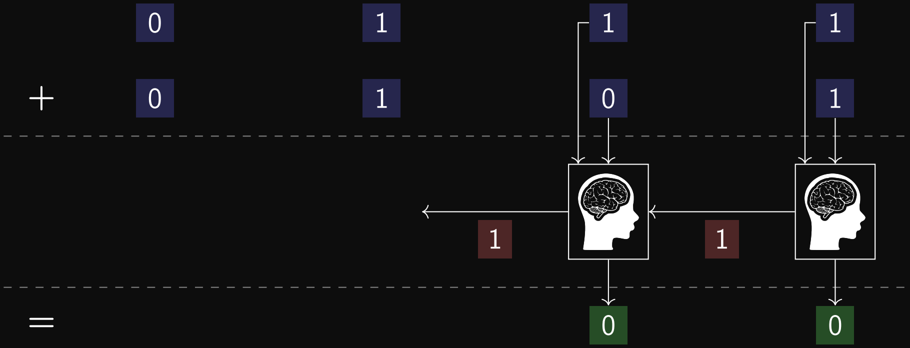
Flattening: Addition



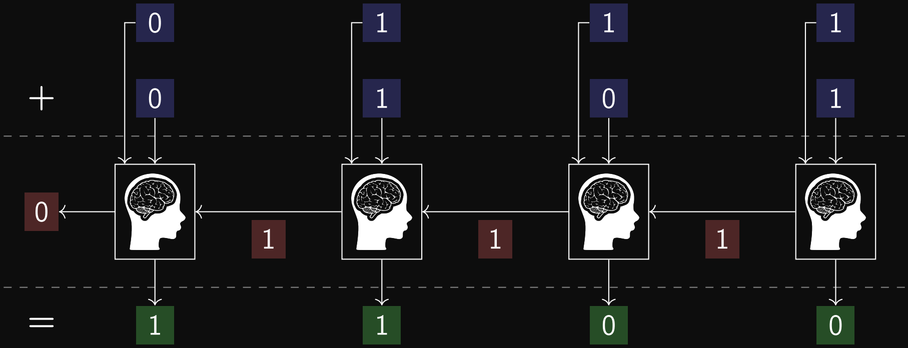
Flattening: Addition



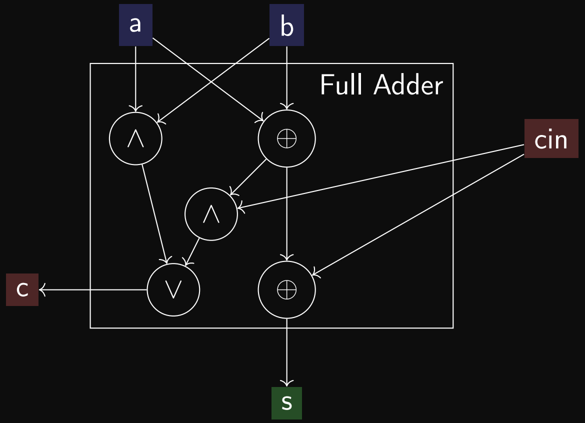
Flattening: Addition



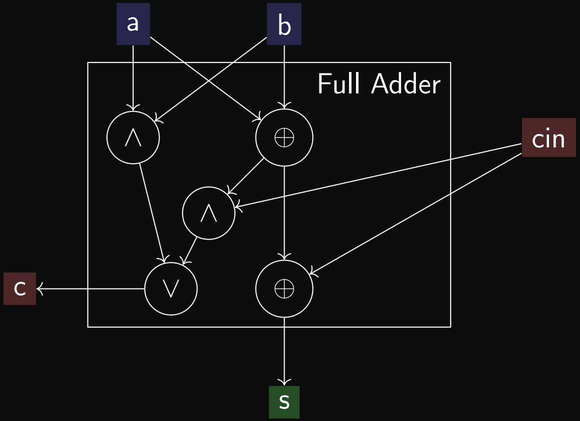
Flattening: Addition



Flattening: Addition



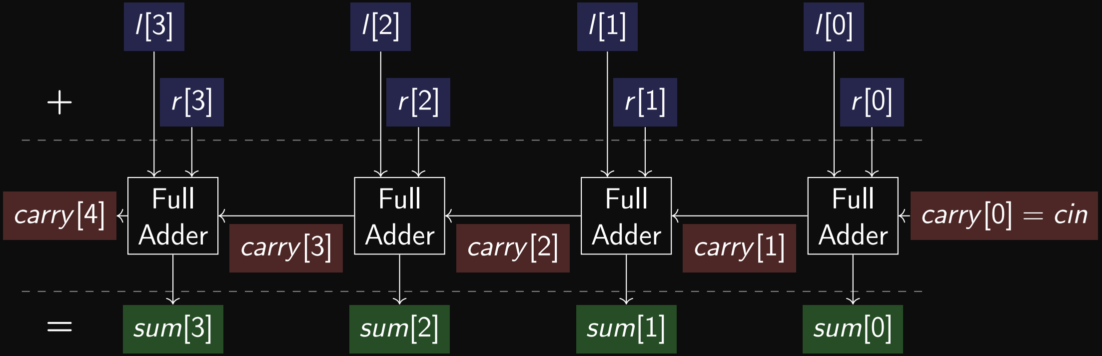
Flattening: Addition



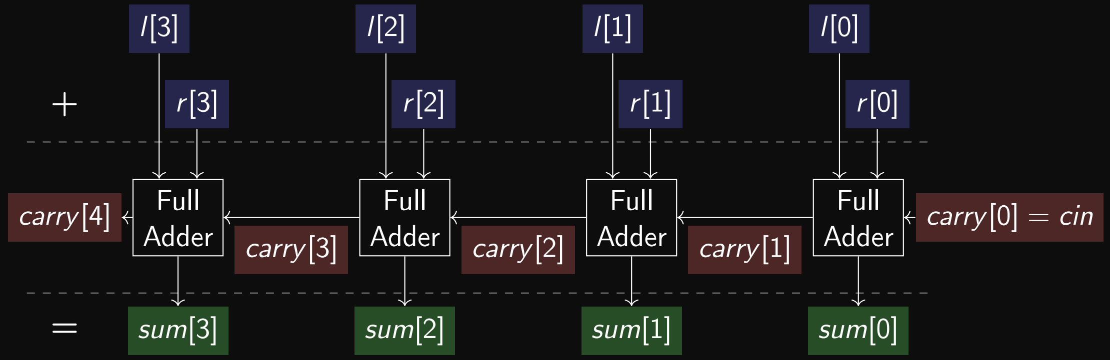
$$s(a, b, cin) = cin \oplus (a \oplus b)$$

$$c(a, b, cin) = (a \wedge b) \vee ((a \oplus b) \wedge cin)$$

Flattening: Addition



Flattening: Addition

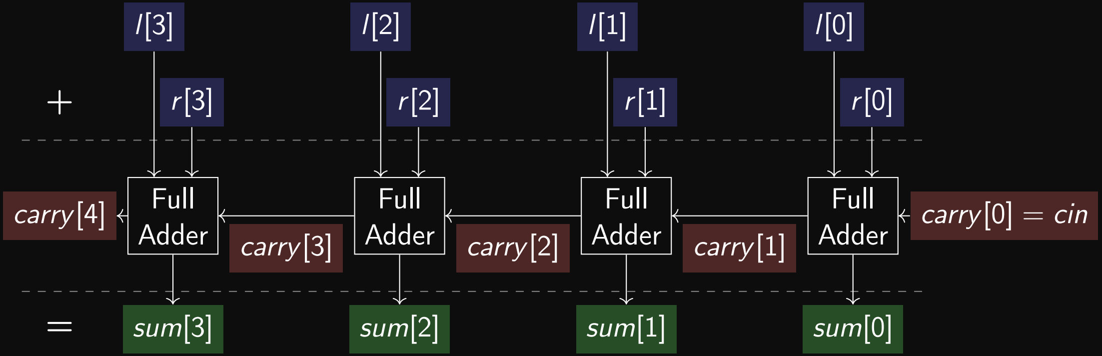


$$\mathrm{carry}(l, r, cin)[0] = cin$$

$$\mathrm{carry}(l, r, cin)[i+1] = \mathrm{c}(l[i+1], r[i+1], \mathrm{carry}(cin, l, r)[i])$$

$$\mathrm{sum}(l, r, cin)[i] = \mathrm{s}(l[i], r[i], \mathrm{carry}(l, r, cin)[i])$$

Flattening: Addition



$$\mathrm{carry}(l, r, cin)[0] = cin$$

$$\mathrm{carry}(l, r, cin)[i+1] = \mathrm{c}(l[i+1], r[i+1], \mathrm{carry}(cin, l, r)[i])$$

$$(l + r)[i] \iff \mathrm{sum}(l, r, cin)[i] = \mathrm{s}(l[i], r[i], \mathrm{carry}(l, r, cin)[i])$$

Flattening: Addition

$$(l + r)[0] \iff \mathsf{s}(l[0], r[0], cin)$$

Flattening: Addition

$$(l + r)[0] \iff \mathsf{s}(l[0], r[0], cin)$$
$$(l + r)[1] \iff \mathsf{s}(l[1], r[1], \mathsf{c}(l[0], r[0], cin))$$

Flattening: Addition

$$(l + r)[0] \iff \mathsf{s}(l[0], r[0], cin)$$
$$(l + r)[1] \iff \mathsf{s}(l[1], r[1], \mathsf{c}(l[0], r[0], cin))$$
$$(l + r)[2] \iff \mathsf{s}(l[2], r[2], \mathsf{c}(l[1], r[1], \mathsf{c}(l[0], r[0], cin)))$$

Flattening: Addition

$$(l + r)[0] \iff \mathsf{s}(l[0], r[0], cin)$$
$$(l + r)[1] \iff \mathsf{s}(l[1], r[1], \mathsf{c}(l[0], r[0], cin))$$
$$(l + r)[2] \iff \mathsf{s}(l[2], r[2], \mathsf{c}(l[1], r[1], \mathsf{c}(l[0], r[0], cin)))$$
$$(l + r)[3] \iff \mathsf{s}(l[3], r[3], \mathsf{c}(l[2], r[2], \mathsf{c}(l[1], r[1], \mathsf{c}(l[0], r[0], cin))))$$

Flattening: Addition

$$(l + r)[0] \iff \mathsf{s}(l[0], r[0], cin)$$
$$(l + r)[1] \iff \mathsf{s}(l[1], r[1], \mathsf{c}(l[0], r[0], cin))$$
$$(l + r)[2] \iff \mathsf{s}(l[2], r[2], \mathsf{c}(l[1], r[1], \mathsf{c}(l[0], r[0], cin)))$$
$$(l + r)[3] \iff \mathsf{s}(l[3], r[3], \mathsf{c}(l[2], r[2], \mathsf{c}(l[1], r[1], \mathsf{c}(l[0], r[0], cin))))$$

$$O(n^2)$$

Flattening: Addition

Auxiliary carry bit vector variable k:

$$k[i+1] \iff \mathrm{c}(l[i], r[i], k[i])$$

Flattening: Addition

Auxiliary carry bit vector variable k:

$$k[i+1] \iff \mathsf{c}(l[i], r[i], k[i])$$
$$(l+r)[i] \iff \mathsf{s}(l[i], r[i], k[i])$$

Flattening: Addition

Auxiliary carry bit vector variable k:

$$k[i+1] \iff \mathsf{c}(l[i], r[i], k[i])$$
$$(l+r)[i] \iff \mathsf{s}(l[i], r[i], k[i])$$

$$k[0] \iff cin \qquad (l+r)[0] \iff \mathsf{s}(l[0], r[0], k[0])$$
$$k[1] \iff \mathsf{c}(l[0], r[0], k[0]) \qquad (l+r)[1] \iff \mathsf{s}(l[1], r[1], k[1])$$
$$k[2] \iff \mathsf{c}(l[1], r[1], k[1]) \qquad (l+r)[2] \iff \mathsf{s}(l[2], r[2], k[2])$$
$$k[3] \iff \mathsf{c}(l[2], r[2], k[2]) \qquad (l+r)[3] \iff \mathsf{s}(l[3], r[3], k[3])$$

Flattening: Addition

Auxiliary carry bit vector variable k:

$$k[i+1] \iff \mathrm{c}(l[i], r[i], k[i])$$
$$(l+r)[i] \iff \mathrm{s}(l[i], r[i], k[i])$$

$$k[0] \iff cin \qquad (l+r)[0] \iff \mathrm{s}(l[0], r[0], k[0])$$
$$k[1] \iff \mathrm{c}(l[0], r[0], k[0]) \qquad (l+r)[1] \iff \mathrm{s}(l[1], r[1], k[1])$$
$$k[2] \iff \mathrm{c}(l[1], r[1], k[1]) \qquad (l+r)[2] \iff \mathrm{s}(l[2], r[2], k[2])$$
$$k[3] \iff \mathrm{c}(l[2], r[2], k[2]) \qquad (l+r)[3] \iff \mathrm{s}(l[3], r[3], k[3])$$

$$O(n)$$

Flattening: Subtraction

$$a = l - r$$

Flattening: Subtraction

$$a = l - r$$

$$\downarrow$$

$$a = l + (-r)$$

Flattening: Subtraction

$$a = l - r$$

$$\downarrow$$

$$a = l + (-r)$$

Two's Complement: $-x = \sim x + 1$

Flattening: Subtraction

$$
\begin{array}{l}
a = l - r \\
\big\downarrow \\
a = l + (-r) \\
\big\downarrow \; \text{Two's Complement: } -x = \sim x + 1 \\
a = l + \sim r + 1
\end{array}
$$

Flattening: Subtraction

$$
\begin{array}{l}
a = l - r \\
\Big\downarrow \\
a = l + (-r) \\
\Big\downarrow \text{Two's Complement: } -x = \sim x + 1 \\
a = l + \sim r + 1
\end{array}
$$

Incoming Carry allows +1 for free

Flattening: Subtraction

$$
\begin{array}{l}
a = l - r \\
\big\downarrow \\
a = l + (-r) \\
\big\downarrow \;\text{Two's Complement: } -x = \sim x + 1 \\
a = l + \sim r + 1 \\
\Big\Downarrow \;\text{Incoming Carry allows } +1 \text{ for free}
\end{array}
$$

$$(l - r)[i] \iff \mathrm{sum}(l, \sim r, 1)[i]$$

Flattening: Comparison

(given: $sz = \text{size}(l) = \text{size}(r)$)

Unsigned:

$$l < r$$

Flattening: Comparison

(given: $sz = \text{size}(l) = \text{size}(r)$)

Unsigned:

$$\begin{array}{c} l < r \\ \downarrow \\ (l - r) < 0 \end{array}$$

Flattening: Comparison

(given: $sz = \text{size}(l) = \text{size}(r)$)

Unsigned:

$$
\begin{array}{c}
l < r \\
\downarrow \\
(l - r) < 0 \\
\Big\downarrow \text{Extension to } sz + 1 \text{ bits}
\end{array}
$$

Flattening: Comparison

(given: $sz = \text{size}(l) = \text{size}(r)$)

Unsigned:

$$
\begin{array}{c}
l < r \\
\downarrow \\
(l - r) < 0 \\
\downarrow \text{Extension to } sz + 1 \text{ bits} \\
\downarrow \text{Perform Subtraction}
\end{array}
$$

Flattening: Comparison

(given: $sz = \text{size}(l) = \text{size}(r)$)

Unsigned:

$$l < r$$
$$\downarrow$$
$$(l - r) < 0$$
$$\Big\downarrow \text{Extension to } sz + 1 \text{ bits}$$
$$\Big\downarrow \text{Perform Subtraction}$$
$$\Big\downarrow \text{Highest Bit is Sign}$$

Flattening: Comparison

(given: $sz = \text{size}(l) = \text{size}(r)$)

Unsigned:

$$
\begin{array}{c}
l < r \\
\big\downarrow \\
(l - r) < 0 \\
\big\downarrow \text{Extension to } sz + 1 \text{ bits} \\
\big\downarrow \text{Perform Subtraction} \\
\big\downarrow \text{Highest Bit is Sign} \\
l < r \iff \text{carry}(l, \sim r, 1)[sz]
\end{array}
$$

Flattening: Comparison

(given: $sz = \text{size}(l) = \text{size}(r)$)

Unsigned:

$$
\begin{array}{c}
l < r \\
\Big\downarrow \\
(l - r) < 0 \\
\Big\downarrow \text{Extension to } sz + 1 \text{ bits} \\
\Big\downarrow \text{Perform Subtraction} \\
\Big\downarrow \text{Highest Bit is Sign} \\
l < r \iff \text{carry}(l, \sim r, 1)[sz]
\end{array}
$$

Signed (sign-extension instead of zero-extension):

$$l < r \iff \text{carry}(l, \sim r, 1)[sz] \oplus (l[sz-1] \iff r[sz-1])$$

Flattening: Static Shift

$$(I \ll_{static} C)[i]$$

Flattening: Static Shift

$$(I \ll_{static} C)[i] = \begin{cases} I[i - C] \\ \end{cases}$$

Flattening: Static Shift

$$(I \ll_{static} C)[i] = \begin{cases} I[i - C] & \text{if } i - C \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Flattening: Static Shift

$$(I \ll_{static} C)[i] = \begin{cases} I[i - C] & \text{if } i - C \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

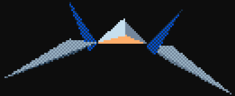
$$(I \gg_{static} C)[i] = \begin{cases} I[i + C] & \text{if } i + C < \text{size}(I) \\ 0 & \text{otherwise} \end{cases}$$

Flattening: Dynamic Shift

7	6	5	4	3	2	1	0		2	1	0
0	0	1	0	1	0	1	0	$\ll$	0	1	0

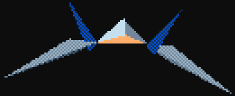
l   r

Flattening: Dynamic Shift



$$\underbrace{\overset{2^w}{\overbrace{\begin{array}{cccccccc} 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ \hline \multicolumn{1}{|c|}{0} & \multicolumn{1}{c|}{0} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{0} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{0} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{0} \\ \hline \end{array}}}}_{\mathsf{l}} \ll \underbrace{\overset{w}{\overbrace{\begin{array}{ccc} 2 & 1 & 0 \\ \hline \multicolumn{1}{|c|}{0} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{0} \\ \hline \end{array}}}}_{\mathsf{r}}$$

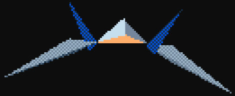
Flattening: Dynamic Shift



$$\underbrace{\overbrace{\begin{array}{cccccccc} 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ \hline \multicolumn{1}{|c|}{0} & \multicolumn{1}{c|}{0} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{0} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{0} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{0} \\ \hline \end{array}}^{2^w}}_{\mathrm{l}} \ll \underbrace{\overbrace{\begin{array}{ccc} 2 & 1 & 0 \\ \hline \multicolumn{1}{|c|}{0} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{0} \\ \hline \end{array}}^{w}}_{\mathrm{r}}$$

$$\text{lshift}(l, r, -1)[i] = l[i]$$

Flattening: Dynamic Shift

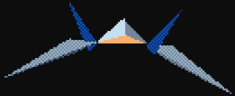


$$\underset{l}{\overbrace{\begin{array}{cccccccc} 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ \hline \multicolumn{1}{|c|}{0} & \multicolumn{1}{c|}{0} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{0} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{0} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{0} \\ \hline \end{array}}^{2^w}} \ll \underset{r}{\overbrace{\begin{array}{ccc} 2 & 1 & 0 \\ \hline \multicolumn{1}{|c|}{0} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{0} \\ \hline \end{array}}^{w}}$$

$$\mathrm{lshift}(l, r, -1)[i] = l[i]$$

$$\mathrm{lshift}(l, r, s)[i] = \Bigg\{$$

Flattening: Dynamic Shift



$$\underset{l}{\overbrace{\begin{array}{|c|c|c|c|c|c|c|c|} \multicolumn{1}{c}{7} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{5} & \multicolumn{1}{c}{4} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{1} & \multicolumn{1}{c}{0} \\ \hline 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ \hline \end{array}}^{2^w}} \ll \underset{r}{\overbrace{\begin{array}{|c|c|c|} \multicolumn{1}{c}{2} & \multicolumn{1}{c}{1} & \multicolumn{1}{c}{0} \\ \hline 0 & 1 & 0 \\ \hline \end{array}}^{w}}$$

$$\mathrm{lshift}(l, r, -1)[i] = l[i]$$

$$\mathrm{lshift}(l, r, s)[i] = \begin{cases} \mathrm{lshift}(l, r, s-1) & \\ & \end{cases}$$

Flattening: Dynamic Shift

$$\overbrace{\begin{array}{|c|c|c|c|c|c|c|c|} \multicolumn{1}{c}{7} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{5} & \multicolumn{1}{c}{4} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{1} & \multicolumn{1}{c}{0} \\ \hline 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ \hline \end{array}}^{2^w}_{\text{l}} \ll \overbrace{\begin{array}{|c|c|c|} \multicolumn{1}{c}{2} & \multicolumn{1}{c}{1} & \multicolumn{1}{c}{0} \\ \hline 0 & 1 & 0 \\ \hline \end{array}}^{w}_{\text{r}}$$

$$\text{lshift}(l, r, -1)[i] = l[i]$$

$$\text{lshift}(l, r, s)[i] = \begin{cases} \text{lshift}(l, r, s-1) & \text{if } r[s] \\ & \end{cases}$$

Flattening: Dynamic Shift

$$\underbrace{\overbrace{\begin{array}{cccccccc} 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ \hline \multicolumn{1}{|c|}{0} & \multicolumn{1}{c|}{0} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{0} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{0} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{0} \\ \hline \end{array}}^{2^w}}_{\mathrm{l}} \ll \underbrace{\overbrace{\begin{array}{ccc} 2 & 1 & 0 \\ \hline \multicolumn{1}{|c|}{0} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{0} \\ \hline \end{array}}^{w}}_{\mathrm{r}}$$

$$\mathrm{lshift}(l, r, -1)[i] = l[i]$$

$$\mathrm{lshift}(l, r, s)[i] = \begin{cases} (\,\mathrm{lshift}(l, r, s-1) \ll_{static} s)[i] & \text{if } r[s] \\ & \end{cases}$$

Flattening: Dynamic Shift

$$\underbrace{\overbrace{\begin{array}{cccccccc} 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ \hline \multicolumn{1}{|c|}{0} & \multicolumn{1}{c|}{0} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{0} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{0} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{0} \\ \hline \end{array}}^{2^w}}_{\mathrm{l}} \ll \underbrace{\overbrace{\begin{array}{ccc} 2 & 1 & 0 \\ \hline \multicolumn{1}{|c|}{0} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{0} \\ \hline \end{array}}^{w}}_{\mathrm{r}}$$

$$\mathrm{lshift}(l, r, -1)[i] = l[i]$$

$$\mathrm{lshift}(l, r, s)[i] = \begin{cases} (\,\mathrm{lshift}(l, r, s-1) \ll_{static} s)[i] & \text{if } r[s] \\ \mathrm{lshift}(l, r, s-1)[i] & \text{if } \neg r[s] \end{cases}$$

Flattening: Dynamic Shift

$$\underbrace{\overbrace{\begin{array}{cccccccc} 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ \hline \boxed{0} & \boxed{0} & \boxed{1} & \boxed{0} & \boxed{1} & \boxed{0} & \boxed{1} & \boxed{0} \end{array}}^{2^w}}_{\text{l}} \ll \underbrace{\overbrace{\begin{array}{ccc} 2 & 1 & 0 \\ \hline \boxed{0} & \boxed{1} & \boxed{0} \end{array}}^{w}}_{\text{r}}$$

$$\text{lshift}(l, r, -1)[i] = l[i]$$

$$\text{lshift}(l, r, s)[i] = \begin{cases} (\,\text{lshift}(l, r, s-1) \ll_{static} s)[i] & \text{if } r[s] \\ \text{lshift}(l, r, s-1)[i] & \text{if } \neg r[s] \end{cases}$$

$$(l \ll r)[i] \iff \text{lshift}(l, r, \text{size}(r) - 1)[i]$$

Flattening: Multiplication

$$l \cdot r =$$

Flattening: Multiplication

$$l \cdot r = l \cdot ((r \ \& \ 1) + (r \ \& \ 10) + (r \ \& \ 100) + \ldots)$$

Flattening: Multiplication

$$\begin{aligned} l \cdot r = l \cdot ((r \mathbin{\&} 1) + (r \mathbin{\&} 10) + (r \mathbin{\&} 100) + \ldots) \\ l \cdot (r \mathbin{\&} 1) \\ + l \cdot (r \mathbin{\&} 10) \\ + l \cdot (r \mathbin{\&} 100) \\ + \ldots \end{aligned}$$

Flattening: Multiplication

$$
\begin{aligned}
l \cdot r = l \cdot ((r \mathbin{\&} 1) + (r \mathbin{\&} 10) + (r \mathbin{\&} 100) + \ldots) \\
l \cdot (r \mathbin{\&} 1) \\
+\, l \cdot (r \mathbin{\&} 10) \\
+\, l \cdot (r \mathbin{\&} 100) \\
+\, \ldots
\end{aligned}
$$

$$\mathrm{mul}(l, r, -1) = 0$$

Flattening: Multiplication

$$
\begin{aligned}
l \cdot r = l \cdot ((r \ \& \ 1) + (r \ \& \ 10) + (r \ \& \ 100) + \ldots) \\
&\quad l \cdot (r \ \& \ 1) \\
&+ l \cdot (r \ \& \ 10) \\
&+ l \cdot (r \ \& \ 100) \\
&+ \ldots
\end{aligned}
$$

$$\mathrm{mul}(l, r, -1) = 0$$

$$\mathrm{mul}(l, r, s) = \ \mathrm{mul}(l, r, s-1) +$$

Flattening: Multiplication

$$
\begin{aligned}
l \cdot r = l \cdot ((r \ \& \ 1) + (r \ \& \ 10) + (r \ \& \ 100) + \ldots) \\
& l \cdot (r \ \& \ 1) \\
&+ l \cdot (r \ \& \ 10) \\
&+ l \cdot (r \ \& \ 100) \\
&+ \ldots
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{mul}(l, r, -1) &= 0 \\
\mathrm{mul}(l, r, s) &= \mathrm{mul}(l, r, s-1) + \begin{cases} (l \ll_{static} s) & \text{if } r[s] \\ 0 & \text{if } \neg r[s] \end{cases}
\end{aligned}
$$

Flattening: Multiplication

$$
\begin{aligned}
l \cdot r = l \cdot ((r \ \& \ 1) + (r \ \& \ 10) + (r \ \& \ 100) + \ldots) \\
l \cdot (r \ \& \ 1) \\
+ \, l \cdot (r \ \& \ 10) \\
+ \, l \cdot (r \ \& \ 100) \\
+ \, \ldots
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{mul}(l, r, -1) &= 0 \\
\mathrm{mul}(l, r, s) &= \ \mathrm{mul}(l, r, s-1) + \begin{cases} (l \ll_{static} s) & \text{if } r[s] \\ 0 & \text{if } \neg r[s] \end{cases} \\
(l \cdot r)[i] &\iff \ \mathrm{mul}(l, r, \mathrm{size}(r) - 1)[i]
\end{aligned}
$$

Flattening: Division (Unsigned)

$$a = l / r$$

Flattening: Division (Unsigned)

$$a = l / r$$

$$\downarrow$$

$$l = a \cdot r + rem$$

Flattening: Division (Unsigned)

$$a = l/r$$

$$\downarrow$$

$$l = a \cdot r + rem$$

$$rem < r$$

Flattening: Division (Unsigned)

$$a = l/r$$

$$\downarrow$$

$$l = a \cdot r + rem$$

$$rem < r$$

$$4_{u8} = 172_{u8} \cdot 3_{u8} + 0_{u8}$$

$$0_{u8} < 3_{u8}$$

Flattening: Division (Unsigned)

$$a = l/r$$

$$\downarrow$$

$$l = a \cdot r + rem$$
$$rem < r$$

$$4_{u8} = 172_{u8} \cdot 3_{u8} + 0_{u8}$$
$$0_{u8} < 3_{u8}$$
$$4_{u8}/3_{u8} = 172_{u8} \lightning$$

Flattening: Division (Unsigned)

$$a = l/r$$

$$\downarrow$$

Extend to $2 \cdot \text{size}$

$$l = a \cdot r + rem$$

$$rem < r$$

Flattening: Division (Unsigned)

$$a = l/r$$

$$\downarrow$$

Extend to $2 \cdot$ size

$$l = a \cdot r + rem$$

$$rem < r$$

✓

Flattening: Division (Signed)

$$a = l/r$$

$$\downarrow$$

Extend to $2 \cdot \text{size}$

$$l = a \cdot r + rem$$

Flattening: Division (Signed)

$$a = l/r$$

$$\downarrow$$

Extend to $2 \cdot$ size

$$l = a \cdot r + rem$$

$$|rem| < |r|$$

Flattening: Division (Signed)

$$a = l / r$$

$$\downarrow$$

Extend to $2 \cdot \text{size}$

$$l = a \cdot r + rem$$

$$|rem| < |r|$$

$$|x| = \begin{cases} \sim x + 1 & \text{if } x[\text{size}(x) - 1] \\ x & \text{otherwise} \end{cases}$$

Flattening: Division (Signed)

$$a = l/r$$

$$\downarrow$$

Extend to $2 \cdot$ size

$$l = a \cdot r + rem$$

$$|rem| < |r|$$

$$-1 = -1 \cdot 2 + 1$$

$$|1| < |2|$$

Flattening: Division (Signed)

$$a = l/r$$

$$\downarrow$$

Extend to $2 \cdot \text{size}$

$$l = a \cdot r + rem$$

$$|rem| < |r|$$

$$-1 = -1 \cdot 2 + 1$$

$$|1| < |2|$$

$$-1/2 = -1 \lightning$$

Flattening: Division (Signed)

$$a = l/r$$

$$\downarrow$$

$$\text{Extend to } 2 \cdot \text{size}$$

$$l = a \cdot r + rem$$

$$|rem| < |r|$$

$$(rem[\text{size}(rem) - 1] \iff l[\text{size}(l - 1)]) \vee \neg \bigwedge_i rem[i]$$

Flattening: Division (Signed)

$$a = l/r$$

$$\downarrow$$

Extend to $2 \cdot \text{size}$

$$l = a \cdot r + rem$$

$$|rem| < |r|$$

$$(rem[\text{size}(rem) - 1] \iff l[\text{size}(l - 1)]) \vee \neg \bigwedge_i rem[i]$$

$$\checkmark$$

Demo

Incremental Flattening

$$(x < y) \land (y < x) \land (a \cdot b = c) \land (b \cdot a = c)$$

Incremental Flattening

$$(x < y) \land (y < x) \land (a \cdot b = c) \land (b \cdot a = c)$$

$(x < y)$

Incremental Flattening

$$(x < y) \wedge (y < x) \wedge (a \cdot b = c) \wedge (b \cdot a = c)$$

$(x < y)$ $\quad$ $x = 0; y = 1$

Incremental Flattening

$$(x < y) \land (y < x) \land (a \cdot b = c) \land (b \cdot a = c)$$

$(x < y)$ $\quad x = 0; y = 1 \implies$ conflict: $(y < x)$

Incremental Flattening

$$(x < y) \wedge (y < x) \wedge (a \cdot b = c) \wedge (b \cdot a = c)$$

$(x < y)$

$(x < y) \wedge (y < x)$

$x = 0; y = 1 \implies$ conflict: $(y < x)$

Incremental Flattening

$$(x < y) \wedge (y < x) \wedge (a \cdot b = c) \wedge (b \cdot a = c)$$

$(x < y)$

$(x < y) \wedge (y < x)$

$x = 0; y = 1 \implies$ conflict: $(y < x)$

Unsatisfiable ✓

Incremental Flattening

$$(x < y) \wedge (y < x) \wedge (a \cdot b = c) \wedge (b \cdot a = c)$$

$(x < y)$	$x = 0; y = 1$ $\Longrightarrow$ conflict: $(y < x)$
$(x < y) \wedge (y < x)$	Unsatisfiable $\checkmark$
$(x < y) \wedge (y < x) \wedge (a \cdot b = c)$	Unsatisfiable
$(x < y) \wedge (y < x) \wedge (a \cdot b = c) \wedge (b \cdot a = c)$	Unsatisfiable

Incremental Flattening

```
incrementalSAT :: [Propositional.Formula] -> [Propositional.Formula]
               -> Propositional.SolveResult
incrementalSAT current pending =
```

Incremental Flattening

```
incrementalSAT :: [Propositional.Formula] -> [Propositional.Formula]
                  -> Propositional.SolveResult
incrementalSAT current pending =
    case SAT.MiniSat.solve current of
```

Incremental Flattening

```
incrementalSAT :: [Propositional.Formula] -> [Propositional.Formula]
                  -> Propositional.SolveResult
incrementalSAT current pending =
    case SAT.MiniSat.solve current of
        Unsatisfiable -> Unsatisfiable
```

Incremental Flattening

```
incrementalSAT :: [Propositional.Formula] -> [Propositional.Formula]
                  -> Propositional.SolveResult
incrementalSAT current pending =
    case SAT.MiniSat.solve current of
        Unsatisfiable -> Unsatisfiable
        Solution assignment ->
            let conflicts =
                filter (λconstraint -> ¬ eval constraint assignment) pending in
```

Incremental Flattening

```
incrementalSAT :: [Propositional.Formula] -> [Propositional.Formula]
                  -> Propositional.SolveResult
incrementalSAT current pending =
    case SAT.MiniSat.solve current of
        Unsatisfiable -> Unsatisfiable
        Solution assignment ->
            let conflicts =
                filter (λconstraint -> ¬ eval constraint assignment) pending in
            if conflicts == [] then
                Solution assignment
```

Incremental Flattening

```
incrementalSAT :: [Propositional.Formula] -> [Propositional.Formula]
                  -> Propositional.SolveResult
incrementalSAT current pending =
    case SAT.MiniSat.solve current of
        Unsatisfiable -> Unsatisfiable
        Solution assignment ->
            let conflicts =
                filter (λconstraint -> ¬ eval constraint assignment) pending in
            if conflicts == [] then
                Solution assignment
            else
                let new = conflicts[0] in
                               (current + new) (pending - new)
```

Incremental Flattening

```
incrementalSAT :: [Propositional.Formula] -> [Propositional.Formula]
                  -> Propositional.SolveResult
incrementalSAT current pending =
    case SAT.MiniSat.solve current of
        Unsatisfiable -> Unsatisfiable
        Solution assignment ->
            let conflicts =
                filter (λconstraint -> ¬ eval constraint assignment) pending in
            if conflicts == [] then
                Solution assignment
            else
                let new = conflicts[0] in
                incrementalSAT (current + new) (pending - new)
```

Incremental Flattening

```
costEstimate :: Propositional.Formula -> Word

solveFlattenedIncremental :: FlattenedFormula -> SolveResult
solveFlattenedIncremental flat =
```

Incremental Flattening

```
costEstimate :: Propositional.Formula -> Word

solveFlattenedIncremental :: FlattenedFormula -> SolveResult
solveFlattenedIncremental flat =
    let initialFormulas = [skeletonOf flat]
        incrementalFormulas = sortOn costEstimate (allConstraints flat)
```

Incremental Flattening

```haskell
costEstimate :: Propositional.Formula -> Word

solveFlattenedIncremental :: FlattenedFormula -> SolveResult
solveFlattenedIncremental flat =
    let initialFormulas = [skeletonOf flat]
        incrementalFormulas = sortOn costEstimate (allConstraints flat)
        satSolution = incrementalSAT initialFormulas incrementalFormulas
```

Incremental Flattening

```
costEstimate :: Propositional.Formula -> Word

solveFlattenedIncremental :: FlattenedFormula -> SolveResult
solveFlattenedIncremental flat =
    let initialFormulas = [skeletonOf flat]
        incrementalFormulas = sortOn costEstimate (allConstraints flat)
        satSolution = incrementalSAT initialFormulas incrementalFormulas
    in reconstructResult satSolution
```

Demo


MY FRIENDS WHEN
I TELL THEM I CAN SMT
CAN YOU
HACK SOMEONE

Extensions

Extensions

Fixed Point

Extensions

Fixed Point

Floating Point

Extensions

Fixed Point

Floating Point

Optimization, e.g. Constant Folding

Extensions

Fixed Point

Floating Point

Optimization, e.g. Constant Folding

Uninterpreted Functions

Applications

https://github.com/Z3Prover/z3/blob/master/src/smt/theory_bv.cpp

Applications: Symbolic Execution

https://klee.github.io

Applications: Symbolic Execution

https://klee.github.io

https://angr.io

https://github.com/thestr4ng3r/shida