```
;
; File :        Anrd.events
;
; Author:       Diederik Verkest
;
; Organization: IMEC, VSDM division
;               Kapeldreef 75,
;               B-3001 Leuven
;               Belgium
;
; Contents:     event file for pc-nqthm to define
;               - the division algorithm Adapted Non Restoring Division (ANRD)
;                 and to prove its correctness with respect to integer
;                 sign-magnitude division.
;               - an implementation of the algorithm on an ALU and the
;                 proof of correctness for the implementation wrt to ANRD.
;
; References:   D. Verkest, L. Claesen, H. De Man, "A Proof of the Non
;               Restoring Division algorithm and its implementation on the
;               Cathedral-II ALU", in Designing Correct Circuits, Eds. J.
;               Staunstrup and R. Sharp, Elseviers Science Publishers B.V.
;               (North-Holland), 1992, pp. 173 - 192
;
;               D. Verkest, L. Claesen, H. De Man, "On the use of the
;               Boyer-Moore theorem prover for correctness proofs of
;               parameterized hardware modules", in Formal VLSI
; Specification and Synthesis (VLSI Design Methods I),
;               Ed. L. J. M. Claesen, Elseviers Science Publishers B.V.
;               (North-Holland), 1990, pp. 99 - 116

;; [Flat file needed; this line removed by Matt K.] (PROVEALL "nrd" '(
```

Event: Start with the initial **nqthm** theory.

Theorem: plus-1
$(1 + x) = (1 + x)$

Theorem: plus-right-id
$(y \notin \mathbf{N}) \rightarrow ((x + y) = \mathrm{fix}(x))$

Theorem: plus-add1
$(x + (1 + y))$
$=$ **if** $y \in \mathbf{N}$ **then** $1 + (x + y)$
else $1 + x$ **endif**

THEOREM: commutativity2-of-plus
$(x + y + z) = (y + x + z)$

THEOREM: commutativity-of-plus
$(x + y) = (y + x)$

THEOREM: associativity-of-plus
$((x + y) + z) = (x + y + z)$

THEOREM: plus-equal-0
$((a + b) = \mathtt{0}) = ((a \simeq \mathtt{0}) \land (b \simeq \mathtt{0}))$

THEOREM: plus-cancellation
$((a + b) = (a + c)) = (\mathrm{fix}\,(b) = \mathrm{fix}\,(c))$

THEOREM: times-zero2
$(y \notin \mathbf{N}) \rightarrow ((x * y) = \mathtt{0})$

THEOREM: distributivity-of-times-over-plus
$(x * (y + z)) = ((x * y) + (x * z))$

THEOREM: times-add1
$(x * (1 + y))$
$= \quad \textbf{if } y \in \mathbf{N} \textbf{ then } x + (x * y)$
$\qquad \textbf{else } \mathrm{fix}\,(x) \textbf{ endif}$

THEOREM: commutativity-of-times
$(x * y) = (y * x)$

THEOREM: commutativity2-of-times
$(x * y * z) = (y * x * z)$

THEOREM: associativity-of-times
$((x * y) * z) = (x * y * z)$

THEOREM: equal-times-0
$((x * y) = \mathtt{0}) = ((x \simeq \mathtt{0}) \lor (y \simeq \mathtt{0}))$

THEOREM: times-1
$(\mathtt{1} * x) = \mathrm{fix}\,(x)$

THEOREM: equal-bools
$(((\mathit{bool1} = \mathbf{t}) \lor (\mathit{bool1} = \mathbf{f})) \land ((\mathit{bool2} = \mathbf{t}) \lor (\mathit{bool2} = \mathbf{f})))$
$\rightarrow \quad ((\mathit{bool1} = \mathit{bool2}) = ((\mathit{bool1} \rightarrow \mathit{bool2}) \land (\mathit{bool2} \rightarrow \mathit{bool1})))$

EVENT: Disable equal-bools.

THEOREM: lessp-times
$((y * x) < (x * z)) = ((x \not\simeq 0) \wedge (y < z))$

THEOREM: times-2-not-1
$(2 * x) \neq 1$

DEFINITION:
$\text{twoto}\,(n)$
$= \quad \textbf{if } n \in \mathbf{N}$
$\quad \textbf{then if } n \simeq 0 \textbf{ then } 1$
$\quad\quad \textbf{else } 2 * \text{twoto}\,(n - 1) \textbf{ endif}$
$\quad \textbf{else } 0 \textbf{ endif}$

THEOREM: twoto-plus
$((j \in \mathbf{N}) \wedge (k \in \mathbf{N})) \rightarrow (\text{twoto}\,(j + k) = (\text{twoto}\,(j) * \text{twoto}\,(k)))$

THEOREM: twoto-by-0
$\text{twoto}\,(0) = 1$

THEOREM: twoto-never-0
$(i \in \mathbf{N}) \rightarrow (0 < \text{twoto}\,(i))$

THEOREM: difference-elim
$((y \in \mathbf{N}) \wedge (y \not< x)) \rightarrow ((x + (y - x)) = y)$

THEOREM: difference-2
$(x - 2) = ((x - 1) - 1)$

THEOREM: difference-x-x
$(x - x) = 0$

THEOREM: difference-plus
$((j + x) - j) = \text{fix}\,(x)$

THEOREM: difference-plus-cancellation
$((a + x) - (a + y)) = (x - y)$

THEOREM: pathological-difference
$(x < y) \rightarrow ((x - y) = 0)$

THEOREM: difference-crock1
$((x + (y - z)) - y)$
$= \quad \textbf{if } y < z \textbf{ then } x - y$
$\quad \textbf{else } x - z \textbf{ endif}$

THEOREM: difference-difference
$((x - y) - z) = (x - (y + z))$

THEOREM: lessp-difference
$((x - y) < x) = ((x \not\simeq \mathsf{0}) \land (y \not\simeq \mathsf{0}))$

THEOREM: difference-add1
$((1 + x) - y)$
$= \ \ \textbf{if } y < (1 + x) \textbf{ then } 1 + (x - y)$
$\quad\quad \textbf{else } \mathsf{0} \textbf{ endif}$

THEOREM: remainder-quotient
$((x \textbf{ mod } y) + (y * (x \div y))) = \text{fix}\,(x)$

THEOREM: remainder-by-nonnumber
$(x \notin \mathbf{N}) \rightarrow ((y \textbf{ mod } x) = \text{fix}\,(y))$

THEOREM: lessp-remainder
$((x \textbf{ mod } y) < y) = (y \not\simeq \mathsf{0})$

THEOREM: remainder-quotient-elim
$((y \not\simeq \mathsf{0}) \land (x \in \mathbf{N})) \rightarrow (((x \textbf{ mod } y) + (y * (x \div y))) = x)$

THEOREM: remainder-x-x
$(x \textbf{ mod } x) = \mathsf{0}$

THEOREM: remainder-plus
$((j + x) \textbf{ mod } j) = (x \textbf{ mod } j)$

THEOREM: remainder-plus-times
$((x + (i * j)) \textbf{ mod } j) = (x \textbf{ mod } j)$

THEOREM: remainder-plus-times-commuted
$((x + (j * i)) \textbf{ mod } j) = (x \textbf{ mod } j)$

THEOREM: remainder-times
$((j * i) \textbf{ mod } j) = \mathsf{0}$

THEOREM: quotient-plus-times
$((x + (i * j)) \div j)$
$= \ \ \textbf{if } j \simeq \mathsf{0} \textbf{ then } \mathsf{0}$
$\quad\quad \textbf{else } i + (x \div j) \textbf{ endif}$

THEOREM: quotient-plus-times-commuted
$((x + (j * i)) \div j)$
$= \ \ \textbf{if } j \simeq \mathsf{0} \textbf{ then } \mathsf{0}$
$\quad\quad \textbf{else } i + (x \div j) \textbf{ endif}$

THEOREM: quotient-times
$((j * i) \div j)$
$=$ **if** $j \simeq \texttt{0}$ **then** $\texttt{0}$
else $\text{fix}\,(i)$ **endif**

EVENT: Disable times.

THEOREM: times-distributes-over-remainder
$((x * y) \textbf{ mod } (x * z)) = (x * (y \textbf{ mod } z))$

THEOREM: remainder-of-1
$(\texttt{1} \textbf{ mod } x)$
$=$ **if** $x = \texttt{1}$ **then** $\texttt{0}$
else $\texttt{1}$ **endif**

THEOREM: remainder-crock3
$((x + (y - z)) \textbf{ mod } y)$
$=$ **if** $(x + (y - z)) < y$ **then** $x + (y - z)$
elseif $y < z$ **then** $x \textbf{ mod } y$
else $(x - z) \textbf{ mod } y$ **endif**

THEOREM: remainder-of-0
$(\texttt{0} \textbf{ mod } x) = \texttt{0}$

THEOREM: remainder-crock4
$((1 + (x + (y - z))) \textbf{ mod } y)$
$=$ **if** $(1 + (x + (y - z))) < y$ **then** $1 + (x + (y - z))$
elseif $y < z$ **then** $(1 + x) \textbf{ mod } y$
elseif $z < (1 + x)$ **then** $(1 + (x - z)) \textbf{ mod } y$
else $\texttt{0}$ **endif**

DEFINITION:
$\text{evenp}\,(n)$
$=$ **if** $n \in \mathbf{N}$
then if $n \simeq \texttt{0}$ **then t**
elseif $n = \texttt{1}$ **then f**
else $\text{evenp}\,((n - 1) - 1)$ **endif**
else f endif

THEOREM: evenp-add1
$((a \in \mathbf{N}) \land \text{evenp}\,(a)) \rightarrow (\neg\ \text{evenp}\,(1 + a))$

THEOREM: not-evenp-add1
$((a \in \mathbf{N}) \land (\neg\ \text{evenp}\,(a))) \rightarrow \text{evenp}\,(1 + a)$

DEFINITION: $\text{exor}(a,\ b) = (((\neg\ a) \wedge b) \vee (a \wedge (\neg\ b)))$

DEFINITION: $\text{boolp}(b) = (\text{truep}(b) \vee \text{falsep}(b))$

THEOREM: lessp-boolp
$\text{boolp}(a < b)$

THEOREM: truep-boolp
$\text{boolp}(a) \rightarrow (\text{truep}(a) = a)$

EVENT: Add the shell *bitvec*, with bottom object function symbol *bv-nil*, with recognizer function symbol *bitvecp*, and 2 accessors: *bv-bit*, with type restriction (one-of truep falsep) and default value false; *bv-vec*, with type restriction (one-of bitvecp) and default value bv-nil.

THEOREM: boolp-bv-bit
$\text{boolp}(\text{bv-bit}(a))$

DEFINITION:
carry (c)
= **if** c **then** 1
else 0 **endif**

DEFINITION: $\text{bv-3}(a) = \text{bv-bit}(a)$

DEFINITION: $\text{bv-2}(a) = \text{bv-bit}(\text{bv-vec}(a))$

DEFINITION: $\text{bv-1}(a) = \text{bv-bit}(\text{bv-vec}(\text{bv-vec}(a)))$

DEFINITION: $\text{bv-0}(a) = \text{bv-bit}(\text{bv-vec}(\text{bv-vec}(\text{bv-vec}(a))))$

DEFINITION:
bv-size (a)
= **if** bitvecp (a)
then if a = BV-NIL **then** 0
else 1 + bv-size (bv-vec (a)) **endif**
else 0 **endif**

THEOREM: size-0
$(\text{bv-size}(x) = 0) = ((x = \text{BV-NIL}) \vee (\neg\ \text{bitvecp}(x)))$

DEFINITION: $\text{controlep}(c) = (\text{bitvecp}(c) \wedge (\text{bv-size}(c) = 4))$

DEFINITION:
bv-invert-even (b)
$=$ **if** bitvecp (b)
then if $b =$ BV-NIL **then** BV-NIL
else bitvec (**if** evenp (bv-size (b)) **then** $\neg$ bv-bit (b)
else bv-bit (b) **endif**,
bv-invert-even (bv-vec (b))) **endif**
else BV-NIL **endif**

DEFINITION:
bv-not (a)
$=$ **if** bitvecp (a)
then if $a =$ BV-NIL **then** BV-NIL
else bitvec ($\neg$ bv-bit (a), bv-not (bv-vec (a))) **endif**
else BV-NIL **endif**

DEFINITION:
bv-exor (a, b)
$=$ **if** bitvecp (a) $\wedge$ bitvecp (b) $\wedge$ (bv-size (a) $=$ bv-size (b))
then if $a =$ BV-NIL **then** BV-NIL
else bitvec (exor (bv-bit (a), bv-bit (b)),
bv-exor (bv-vec (a), bv-vec (b))) **endif**
else BV-NIL **endif**

THEOREM: bv-exor-nil-a
$(a =$ BV-NIL$)$ $\rightarrow$ (bv-exor (a, b) $=$ BV-NIL)

THEOREM: bv-exor-nil-b
$(b =$ BV-NIL$)$ $\rightarrow$ (bv-exor (a, b) $=$ BV-NIL)

DEFINITION:
zero-bitvec (n)
$=$ **if** $n \in \mathbf{N}$
then if $n \simeq 0$ **then** BV-NIL
else bitvec (**f**, zero-bitvec $(n - 1)$) **endif**
else BV-NIL **endif**

DEFINITION:
one-bitvec (n)
$=$ **if** $n \in \mathbf{N}$
then if $n \simeq 0$ **then** BV-NIL
else bitvec (**t**, one-bitvec $(n - 1)$) **endif**
else BV-NIL **endif**

THEOREM: size-zerobv
bv-size (zero-bitvec (bv-size (a))) $=$ bv-size (a)

THEOREM: size-onebv
bv-size (one-bitvec (bv-size (a))) = bv-size (a)

THEOREM: size-bvnot
bv-size (bv-not (a)) = bv-size (a)

THEOREM: size-bvexor
(bitvecp (a) $\wedge$ bitvecp (b) $\wedge$ (bv-size (a) = bv-size (b)))
$\rightarrow$ (bv-size (bv-exor (a, b)) = bv-size (a))

THEOREM: size-bvinv
bv-size (bv-invert-even (a)) = bv-size (a)

DEFINITION:
bv-append (a, b)
= **if** bitvecp (a) $\wedge$ bitvecp (b)
then if a = BV-NIL **then** b
else bitvec (bv-bit (a), bv-append (bv-vec (a), b)) **endif**
else BV-NIL **endif**

EVENT: Add the shell *bitvec-carry-ovf*, with bottom object function symbol *bvco-nil*, with recognizer function symbol *bitvec-carry-ovfp*, and 3 accessors: *bvco-bitvec*, with type restriction (one-of bitvecp) and default value bv-nil; *bvco-carry*, with type restriction (one-of truep falsep) and default value false; *bvco-ovf*, with type restriction (one-of truep falsep) and default value false.

THEOREM: boolp-bvco-carry
boolp (bvco-carry (a))

THEOREM: boolp-bvco-ovf
boolp (bvco-ovf (a))

EVENT: Add the shell *carry-sign-ovf-bitvec*, with bottom object function symbol *csobv-nil*, with recognizer function symbol *carry-sign-ovf-bitvecp*, and 4 accessors: *csobv-carry*, with type restriction (one-of truep falsep) and default value false; *csobv-sign*, with type restriction (one-of truep falsep) and default value false; *csobv-ovf*, with type restriction (one-of truep falsep) and default value false; *csobv-bitvec*, with type restriction (one-of bitvecp) and default value bv-nil.

DEFINITION:
tcp (x) = (($x \in \mathbf{N}$) $\vee$ (negativep (x) $\wedge$ (negative-guts (x) $\not\simeq$ 0)))

DEFINITION:
tc-in-rangep (x, n)
$=$ **if** $n \simeq 0$ **then f**
elseif negativep (x) **then** twoto $(n - 1) \not< $ negative-guts (x)
else $x <$ twoto $(n - 1)$ **endif**

DEFINITION:
add (a, b)
$=$ **if** negativep (a)
then if negativep (b) **then** $-$ (negative-guts (a) $+$ negative-guts (b))
elseif $b <$ negative-guts (a) **then** $-$ (negative-guts (a) $-$ b)
else b $-$ negative-guts (a) **endif**
elseif negativep (b)
then if $a <$ negative-guts (b) **then** $-$ (negative-guts (b) $-$ a)
else a $-$ negative-guts (b) **endif**
else $a + b$ **endif**

THEOREM: commutativity2-of-add
add $(x,$ add $(y, z))$ $=$ add $(y,$ add $(x, z))$

THEOREM: commutativity-of-add
add (x, y) $=$ add (y, x)

THEOREM: associativity-of-add
add (add $(x, y), z)$ $=$ add $(x,$ add $(y, z))$

DEFINITION:
bv-to-nat (a)
$=$ **if** bitvecp (a)
then if $a =$ BV-NIL **then** 0
else (**if** bv-bit (a) **then** 1
else 0 **endif**
$*$ twoto (bv-size $(a) - 1$))
$+$ bv-to-nat (bv-vec (a)) **endif**
else 0 **endif**

DEFINITION:
nat-to-bv (a, size)
$=$ **if** $\mathit{size} \simeq 0$ **then** BV-NIL
else bitvec (**if** $(a \div$ twoto $(\mathit{size} - 1)) = 1$ **then t**
else f endif,
nat-to-bv (a **mod** twoto $(\mathit{size} - 1)$, $\mathit{size} - 1$)) **endif**

DEFINITION:
tc-to-integer (a)

$=$ **if** bitvecp(a)
then if falsep (bv-bit(a)) **then** bv-to-nat(a)
else add (bv-to-nat(a), $-$ twoto (bv-size(a))) **endif**
else `0` **endif**

DEFINITION:
nat-to-integer$(n,\ size)$
$=$ **if** $n <$ twoto$(size - 1)$ **then** n
else $-$ (twoto$(size) - n$) **endif**

DEFINITION:
integer-to-nat$(n,\ size)$
$=$ **if** negativep(n) **then** twoto$(size)$ $-$ negative-guts(n)
else n **endif**

THEOREM: upper-bound-on-bv-to-nat
bv-to-nat$(a) <$ twoto (bv-size(a))

THEOREM: bv-to-nat-to-integer-lemma2
bv-bit(a)
$=$ **if** bv-size$(a) \simeq$ `0` **then f**
else bv-to-nat$(a) \not<$ twoto (bv-size$(a) - 1$) **endif**

THEOREM: nat-to-bv-of-trunc
(bitvecp(a) $\wedge$ boolp(b))
$\rightarrow$ (bv-to-nat(a) = (bv-to-nat (bitvec$(b,\ a)$) **mod** twoto (bv-size(a))))

THEOREM: tcp-tc-to-integer
tcp (tc-to-integer(n))

THEOREM: upper-bound-on-non-negative-bv-to-nat
(bitvecp(a) $\wedge$ $(a \neq$ BV-NIL$)$ $\wedge$ $(\neg$ bv-bit(a)))
$\rightarrow$ (bv-to-nat$(a) <$ twoto (bv-size$(a) - 1$))

THEOREM: lower-bound-on-negative-bv-to-nat
(bitvecp(a) $\wedge$ $(a \neq$ BV-NIL$)$ $\wedge$ bv-bit(a))
$\rightarrow$ (bv-to-nat$(a) \not<$ twoto (bv-size$(a) - 1$))

THEOREM: integer-in-rangep-of-tc-to-integer
$(n =$ bv-size(a))
$\rightarrow$ (tc-in-rangep (tc-to-integer(a), n) = (bitvecp(a) $\wedge$ $(a \neq$ BV-NIL$)$))

THEOREM: plus-to-add
(tcp(x) $\wedge$ tcp(y) $\wedge$ tc-in-rangep$(x,\ n)$ $\wedge$ tc-in-rangep$(y,\ n)$)
$\rightarrow$ (nat-to-integer ((carry(c)
$+$ integer-to-nat$(x,\ n)$

$+$ integer-to-nat $(y,\ n))$
mod twoto (n),
$n)$

$=$ **if** tc-in-rangep (add $(x,$ add $(y,$ carry $(c))),\ n)$
then add $(x,$ add $(y,$ carry $(c)))$
elseif negativep (add $(x,$ add $(y,$ carry $(c))))$
then add $(x,$ add $(y,$ add (carry (c), twoto $(n))))$
else add $(x,$ add $(y,$ add (carry (c), $-$ twoto $(n))))$ **endif**)

THEOREM: times-2-twoto
$(a \in \mathbf{N}) \rightarrow ((2 * \text{twoto}(a)) = \text{twoto}(1 + a))$

EVENT: Disable times-2-twoto.

THEOREM: bv-to-nat-to-integer
tc-to-integer (a) = nat-to-integer (bv-to-nat (a), bv-size (a))

EVENT: Disable bv-to-nat-to-integer.

THEOREM: tc-to-integer-to-nat
$(n =$ bv-size $(a))$
$\rightarrow$ (bv-to-nat (a) = integer-to-nat (tc-to-integer (a), n))

EVENT: Disable tc-to-integer-to-nat.

THEOREM: bit-on-implies-non-0
bv-bit $(a) \rightarrow$ (bv-to-nat $(a) \neq 0$)

DEFINITION:
bv-adder $(a,\ b,\ cin)$
$=$ **if** bitvecp (a)
$\wedge$ bitvecp (b)
$\wedge$ (bv-size (a) = bv-size (b))
$\wedge$ boolp (cin)
then if a = BV-NIL **then** bitvec-carry-ovf (BV-NIL, cin, **f**)
else bitvec-carry-ovf (bitvec (exor (exor (bv-bit (a), bv-bit (b)),
bvco-carry (bv-adder (bv-vec (a),
bv-vec (b),
cin))),
bvco-bitvec (bv-adder (bv-vec (a),
bv-vec (b),
cin))),
(bv-bit $(a) \wedge$ bv-bit (b))

∨ (bv-bit (a)
∧ bvco-carry (bv-adder (bv-vec (a),
bv-vec (b),
cin)))
∨ (bv-bit (b)
∧ bvco-carry (bv-adder (bv-vec (a),
bv-vec (b),
cin))),
bvco-carry (bv-adder (bv-vec (a),
bv-vec (b),
cin))) **endif**

else BVCO-NIL **endif**

THEOREM: size-bv-adder
(bitvecp (a) ∧ bitvecp (b) ∧ boolp (cin) ∧ (bv-size (a) = bv-size (b)))
→ (bv-size (bvco-bitvec (bv-adder (a, b, cin))) = bv-size (a))

THEOREM: bv-adder-plusses
(bitvecp (a) ∧ bitvecp (b) ∧ boolp (cin) ∧ (bv-size (a) = bv-size (b)))
→ (bv-to-nat (bitvec (bvco-carry (bv-adder (a, b, cin)),
bvco-bitvec (bv-adder (a, b, cin))))
= (bv-to-nat (a) + bv-to-nat (b) + carry (cin)))

THEOREM: bv-adder-non-nil
(bitvecp (a)
∧ bitvecp (b)
∧ (bv-size (a) = bv-size (b))
∧ boolp (cin)
∧ (a ≠ BV-NIL)
∧ (b ≠ BV-NIL))
→ ((bvco-bitvec (bv-adder (a, b, cin)) = BV-NIL) = **f**)

THEOREM: nat-interpretation-of-bv-adder-output
(bitvecp (a) ∧ bitvecp (b) ∧ (bv-size (a) = bv-size (b)) ∧ boolp (cin))
→ (bv-to-nat (bvco-bitvec (bv-adder (a, b, cin)))
= ((bv-to-nat (a) + bv-to-nat (b) + carry (cin))
mod twoto (bv-size (a))))

EVENT: Disable nat-interpretation-of-bv-adder-output.

THEOREM: integer-interpretation-of-bv-adder-output-lemma1
(bitvecp (a) ∧ bitvecp (b) ∧ (bv-size (a) = bv-size (b)) ∧ boolp (cin))
→ (tc-to-integer (bvco-bitvec (bv-adder (a, b, cin)))
= nat-to-integer ((bv-to-nat (a) + bv-to-nat (b) + carry (cin))
mod twoto (bv-size (a)),
bv-size (a)))

THEOREM: integer-interpretation-of-bv-adder-output

$$
\begin{array}{l}
(\text{bitvecp}\,(a) \\
\;\wedge\;\; \text{bitvecp}\,(b) \\
\;\wedge\;\; (a \neq \text{BV-NIL}) \\
\;\wedge\;\; \text{boolp}\,(cin) \\
\;\wedge\;\; (b \neq \text{BV-NIL}) \\
\;\wedge\;\; (\text{bv-size}\,(a) = \text{bv-size}\,(b))) \\
\rightarrow\;\; (\text{tc-to-integer}\,(\text{bvco-bitvec}\,(\text{bv-adder}\,(a,\ b,\ cin))) \\
\quad = \;\; \textbf{if}\ \text{tc-in-rangep}\,(\text{add}\,(\text{tc-to-integer}\,(a), \\
\qquad\qquad\qquad\qquad \text{add}\,(\text{tc-to-integer}\,(b),\ \text{carry}\,(cin))), \\
\qquad\qquad\qquad \text{bv-size}\,(a)) \\
\qquad\quad \textbf{then}\ \text{add}\,(\text{tc-to-integer}\,(a),\ \text{add}\,(\text{tc-to-integer}\,(b),\ \text{carry}\,(cin))) \\
\qquad\quad \textbf{elseif}\ \text{negativep}\,(\text{add}\,(\text{tc-to-integer}\,(a), \\
\qquad\qquad\qquad\qquad \text{add}\,(\text{tc-to-integer}\,(b),\ \text{carry}\,(cin)))) \\
\qquad\quad \textbf{then}\ \text{add}\,(\text{tc-to-integer}\,(a), \\
\qquad\qquad\qquad \text{add}\,(\text{tc-to-integer}\,(b), \\
\qquad\qquad\qquad\quad \text{add}\,(\text{carry}\,(cin),\ \text{twoto}\,(\text{bv-size}\,(a))))) \\
\qquad\quad \textbf{else}\ \text{add}\,(\text{tc-to-integer}\,(a), \\
\qquad\qquad\qquad \text{add}\,(\text{tc-to-integer}\,(b), \\
\qquad\qquad\qquad\quad \text{add}\,(\text{carry}\,(cin),\ -\ \text{twoto}\,(\text{bv-size}\,(a))))) \ \textbf{endif})
\end{array}
$$

EVENT: Disable integer-interpretation-of-bv-adder-output-lemma1.

EVENT: Disable integer-interpretation-of-bv-adder-output.

DEFINITION:

$\text{alucod-fc}\,(ik,\ ip,\ cc) = ((ik \wedge (\neg\ ip)) \vee (ip \wedge (\neg\ cc)))$

THEOREM: boolp-alucod-fc

$\text{boolp}\,(\text{alucod-fc}\,(k,\ p,\ c))$

DEFINITION:

$\text{alucev-fc}\,(ik,\ ip,\ cc) = (((\neg\ ik) \wedge (\neg\ ip)) \vee (ip \wedge (\neg\ cc)))$

THEOREM: boolp-alucev-fc

$\text{boolp}\,(\text{alucev-fc}\,(k,\ p,\ c))$

DEFINITION:

$$
\begin{array}{l}
\text{addbyp-fcout}\,(cinshot,\ prop1,\ prop2,\ prop3,\ prop4,\ cinnorm) \\
= \;\; \textbf{if}\ prop1 \wedge prop2 \wedge prop3 \wedge prop4\ \ \textbf{then}\ cinshot \\
\qquad \textbf{else}\ cinnorm\ \textbf{endif}
\end{array}
$$

DEFINITION:

$\text{c4}\,(cp1,\ cp2,\ cp3,\ cp4,\ ck1,\ ck2,\ ck3,\ ck4,\ cin)$

```
=   if boolp (cp1)
      ∧  boolp (cp2)
      ∧  boolp (cp3)
      ∧  boolp (cp4)
      ∧  boolp (ck1)
      ∧  boolp (ck2)
      ∧  boolp (ck3)
      ∧  boolp (ck4)
      ∧  boolp (cin)
    then bitvec-carry-ovf (bitvec (alucod-fc (ck2,
                                              cp2,
                                              alucev-fc (ck3,
                                                         cp3,
                                                         alucod-fc (ck4,
                                                                    cp4,
                                                                    cin))),
                                   bitvec (alucev-fc (ck3,
                                                      cp3,
                                                      alucod-fc (ck4, cp4, cin)),
                                           bitvec (alucod-fc (ck4, cp4, cin),
                                                   bitvec (cin, BV-NIL)))),
                           alucev-fc (ck1,
                                      cp1,
                                      alucod-fc (ck2,
                                                 cp2,
                                                 alucev-fc (ck3,
                                                            cp3,
                                                            alucod-fc (ck4,
                                                                       cp4,
                                                                       cin)))),
                           alucod-fc (ck2,
                                      cp2,
                                      alucev-fc (ck3,
                                                 cp3,
                                                 alucod-fc (ck4, cp4, cin))))
    else BVCO-NIL endif
```

DEFINITION:

```
c4byp (cp1, cp2, cp3, cp4, ck1, ck2, ck3, ck4, cin)
=   if boolp (cp1)
      ∧  boolp (cp2)
      ∧  boolp (cp3)
      ∧  boolp (cp4)
      ∧  boolp (ck1)
```

```
    ∧  boolp (ck2)
    ∧  boolp (ck3)
    ∧  boolp (ck4)
    ∧  boolp (cin)
  then bitvec-carry-ovf (bitvec (alucod-fc (ck2,
                                            cp2,
                                            alucev-fc (ck3,
                                                       cp3,
                                                       alucod-fc (ck4,
                                                                  cp4,
                                                                  cin))),
                          bitvec (alucev-fc (ck3,
                                             cp3,
                                             alucod-fc (ck4, cp4, cin)),
                                  bitvec (alucod-fc (ck4, cp4, cin),
                                          bitvec (cin, BV-NIL)))),
                          addbyp-fcout (cin,
                                        cp4,
                                        cp3,
                                        cp2,
                                        cp1,
                                        alucev-fc (ck1,
                                                   cp1,
                                                   alucod-fc (ck2,
                                                              cp2,
                                                              alucev-fc (ck3,
                                                                         cp3,
                                                                         alucod-fc (ck4,
                                                                                    cp4,
                                                                                    cin))))),
                          alucod-fc (ck2,
                                     cp2,
                                     alucev-fc (ck3,
                                                cp3,
                                                alucod-fc (ck4, cp4, cin))))
  else BVCO-NIL endif
```

THEOREM: c4-c4byp-help

```
(boolp (ck1)
 ∧  boolp (ck2)
 ∧  boolp (ck3)
 ∧  boolp (ck4)
 ∧  boolp (cp1)
 ∧  boolp (cp2)
```

∧ boolp (*cp3*)
∧ boolp (*cp4*)
∧ boolp (*cin*))
→ (addbyp-fcout (*cin*,
cp4,
cp3,
cp2,
cp1,
alucev-fc (*ck1*,
cp1,
alucod-fc (*ck2*,
cp2,
alucev-fc (*ck3*,
cp3,
alucod-fc (*ck4*, *cp4*, *cin*)))))
= alucev-fc (*ck1*,
cp1,
alucod-fc (*ck2*,
cp2,
alucev-fc (*ck3*, *cp3*, alucod-fc (*ck4*, *cp4*, *cin*)))))

THEOREM: c4-c4byp-relate
(boolp (*p1*)
∧ boolp (*p2*)
∧ boolp (*p3*)
∧ boolp (*p4*)
∧ boolp (*k1*)
∧ boolp (*k2*)
∧ boolp (*k3*)
∧ boolp (*k4*)
∧ boolp (*cin*))
→ (c4byp (*p1*, *p2*, *p3*, *p4*, *k1*, *k2*, *k3*, *k4*, *cin*)
= c4 (*p1*, *p2*, *p3*, *p4*, *k1*, *k2*, *k3*, *k4*, *cin*))

EVENT: Disable c4.

EVENT: Disable c4byp.

EVENT: Disable c4-c4byp-help.

DEFINITION:
carry-col (*ik*, *ip*, *ccin*)
= **if** bitvecp (*ik*)

```
        ∧  bitvecp (ip)
        ∧  boolp (ccin)
        ∧  (bv-size (ik) = bv-size (ip))
    then if ik = BV-NIL then bitvec-carry-ovf (BV-NIL, ccin, f)
         else bitvec-carry-ovf (bitvec (bvco-carry (carry-col (bv-vec (ik),
                                                               bv-vec (ip),
                                                               ccin)),
                                       bvco-bitvec (carry-col (bv-vec (ik),
                                                               bv-vec (ip),
                                                               ccin))),
                                if evenp (bv-size (ik))
                                then alucev-fc (bv-bit (ik),
                                                bv-bit (ip),
                                                bvco-carry (carry-col (bv-vec (ik),
                                                                       bv-vec (ip),
                                                                       ccin)))
                                else alucod-fc (bv-bit (ik),
                                                bv-bit (ip),
                                                bvco-carry (carry-col (bv-vec (ik),
                                                                       bv-vec (ip),
                                                                       ccin))) endif,
                                bvco-carry (carry-col (bv-vec (ik),
                                                       bv-vec (ip),
                                                       ccin))) endif
    else BVCO-NIL endif
```

THEOREM: size-carry-col
(bitvecp (a) ∧ bitvecp (b) ∧ (bv-size (a) = bv-size (b)) ∧ boolp (*ccin*))
→ (bv-size (bvco-bitvec (carry-col $(a,\ b,\ ccin)$))) = bv-size (a))

DEFINITION: up (*nb*) = ((4 ∗ ((*nb* − 10) ÷ 4)) + 9)

DEFINITION:

```
carry-col2 (ik, ip, ccin, nb)
=   if bitvecp (ik)
        ∧  bitvecp (ip)
        ∧  boolp (ccin)
        ∧  (bv-size (ik) = bv-size (ip))
        ∧  evenp (nb)
    then if ik = BV-NIL then bitvec-carry-ovf (BV-NIL, ccin, f)
         elseif (bv-size (ik) < up (nb))
                ∧  (11 < nb)
                ∧  (4 < bv-size (ik))
                ∧  evenp (bv-size (ik))
         then bitvec-carry-ovf (bv-append (bvco-bitvec (c4 (bv-bit (ip),
```

```
                                                   bv-bit (bv-vec (ip)),
                                                   bv-bit (bv-vec (bv-vec (ip))),
                                                   bv-bit (bv-vec (bv-vec (bv-vec (ip)))),
                                                   bv-bit (ik),
                                                   bv-bit (bv-vec (ik)),
                                                   bv-bit (bv-vec (bv-vec (ik))),
                                                   bv-bit (bv-vec (bv-vec (bv-vec (ik)))),
                                                   bvco-carry (carry-col2 (bv-vec (bv-vec (bv-vec (bv-
                                                                           bv-vec (bv-vec (bv-vec (bv-
                                                                           ccin,
                                                                           nb)))),
                                    bvco-bitvec (carry-col2 (bv-vec (bv-vec (bv-vec (bv-vec (ik)))),
                                                             bv-vec (bv-vec (bv-vec (bv-vec (ip)))),
                                                             ccin,
                                                             nb))),
                            bvco-carry (c4 (bv-bit (ip),
                                           bv-bit (bv-vec (ip)),
                                           bv-bit (bv-vec (bv-vec (ip))),
                                           bv-bit (bv-vec (bv-vec (bv-vec (ip)))),
                                           bv-bit (ik),
                                           bv-bit (bv-vec (ik)),
                                           bv-bit (bv-vec (bv-vec (ik))),
                                           bv-bit (bv-vec (bv-vec (bv-vec (ik)))),
                                           bvco-carry (carry-col2 (bv-vec (bv-vec (bv-vec (bv-vec (ik)))),
                                                                   bv-vec (bv-vec (bv-vec (bv-vec (ip)))),
                                                                   ccin,
                                                                   nb)))),
                            bvco-carry (carry-col2 (bv-vec (ik),
                                                    bv-vec (ip),
                                                    ccin,
                                                    nb)))
               else bitvec-carry-ovf (bitvec (bvco-carry (carry-col2 (bv-vec (ik),
                                                                      bv-vec (ip),
                                                                      ccin,
                                                                      nb)),
                                         bvco-bitvec (carry-col2 (bv-vec (ik),
                                                                  bv-vec (ip),
                                                                  ccin,
                                                                  nb))),
                            if evenp (bv-size (ik))
                            then alucev-fc (bv-bit (ik),
                                            bv-bit (ip),
                                            bvco-carry (carry-col2 (bv-vec (ik),
                                                                    bv-vec (ip),
```

```
                                                      ccin,
                                                      nb)))
              else alucod-fc (bv-bit (ik),
                              bv-bit (ip),
                              bvco-carry (carry-col2 (bv-vec (ik),
                                                      bv-vec (ip),
                                                      ccin,
                                                      nb))) endif,
              bvco-carry (carry-col2 (bv-vec (ik),
                                      bv-vec (ip),
                                      ccin,
                                      nb))) endif
   else BVCO-NIL endif
```

THEOREM: app-c4-carry1

```
 (bitvecp (ik)
  ∧  bitvecp (ip)
  ∧  boolp (ccin)
  ∧  evenp (bv-size (ik))
  ∧  (bv-size (ik) = bv-size (ip))
  ∧  (4 < bv-size (ik)))
 →  (bv-append (bvco-bitvec (c4 (bv-bit (ip),
                                  bv-bit (bv-vec (ip)),
                                  bv-bit (bv-vec (bv-vec (ip))),
                                  bv-bit (bv-vec (bv-vec (bv-vec (ip)))),
                                  bv-bit (ik),
                                  bv-bit (bv-vec (ik)),
                                  bv-bit (bv-vec (bv-vec (ik))),
                                  bv-bit (bv-vec (bv-vec (bv-vec (ik)))),
                                  bvco-carry (carry-col (bv-vec (bv-vec (bv-vec (bv-vec (ik)))),
                                                         bv-vec (bv-vec (bv-vec (bv-vec (ip)))),
                                                         ccin)))),
                bvco-bitvec (carry-col (bv-vec (bv-vec (bv-vec (bv-vec (ik)))),
                                        bv-vec (bv-vec (bv-vec (bv-vec (ip)))),
                                        ccin)))
     =  bitvec (bvco-carry (carry-col (bv-vec (ik), bv-vec (ip), ccin)),
                bvco-bitvec (carry-col (bv-vec (ik), bv-vec (ip), ccin))))
```

THEOREM: app-c4-carry2

```
 (bitvecp (ik)
  ∧  bitvecp (ip)
  ∧  boolp (ccin)
  ∧  evenp (bv-size (ik))
  ∧  (bv-size (ik) = bv-size (ip))
```

$\wedge$ $(4 < \text{bv-size}(ik)))$
$\rightarrow$ (bvco-carry (c4 (bv-bit (*ip*),
bv-bit (bv-vec (*ip*)),
bv-bit (bv-vec (bv-vec (*ip*))),
bv-bit (bv-vec (bv-vec (bv-vec (*ip*)))),
bv-bit (*ik*),
bv-bit (bv-vec (*ik*)),
bv-bit (bv-vec (bv-vec (*ik*))),
bv-bit (bv-vec (bv-vec (bv-vec (*ik*)))),
bvco-carry (carry-col (bv-vec (bv-vec (bv-vec (bv-vec (*ik*)))),
bv-vec (bv-vec (bv-vec (bv-vec (*ip*)))),
ccin))))
= **if** evenp (bv-size (*ik*))
then alucev-fc (bv-bit (*ik*),
bv-bit (*ip*),
bvco-carry (carry-col (bv-vec (*ik*),
bv-vec (*ip*),
ccin)))
else alucod-fc (bv-bit (*ik*),
bv-bit (*ip*),
bvco-carry (carry-col (bv-vec (*ik*),
bv-vec (*ip*),
ccin))) **endif**)

DEFINITION:
induct-carry-col2 (*ik*, *ip*, *ccin*, *nb*)
= **if** bitvecp (*ik*)
$\wedge$ bitvecp (*ip*)
$\wedge$ boolp (*ccin*)
$\wedge$ (bv-size (*ik*) = bv-size (*ip*))
$\wedge$ evenp (*nb*)
then if *ik* = BV-NIL **then t**
elseif (bv-size (*ik*) < up (*nb*))
$\wedge$ $(11 < nb)$
$\wedge$ (4 < bv-size (*ik*))
$\wedge$ evenp (bv-size (*ik*))
then induct-carry-col2 (bv-vec (bv-vec (bv-vec (bv-vec (*ik*)))),
bv-vec (bv-vec (bv-vec (bv-vec (*ip*)))),
ccin,
nb)
$\wedge$ induct-carry-col2 (bv-vec (*ik*), bv-vec (*ip*), *ccin*, *nb*)
else induct-carry-col2 (bv-vec (*ik*), bv-vec (*ip*), *ccin*, *nb*) **endif**
else t endif

THEOREM: carry2-carry-relate

evenp (nb) → (carry-col2 (ik, ip, $ccin$, nb) = carry-col (ik, ip, $ccin$))

DEFINITION:
cbyp-col (ik, ip, $ccin$, nb)
= **if** bitvecp (ik)
∧ bitvecp (ip)
∧ boolp ($ccin$)
∧ (bv-size (ik) = bv-size (ip))
∧ evenp (nb)
then if ik = BV-NIL **then** bitvec-carry-ovf (BV-NIL, $ccin$, **f**)
elseif (bv-size (ik) < up (nb))
∧ (11 < nb)
∧ (4 < bv-size (ik))
∧ evenp (bv-size (ik))
then bitvec-carry-ovf (bv-append (bvco-bitvec (c4byp (bv-bit (ip),
bv-bit (bv-vec (ip)),
bv-bit (bv-vec (bv-vec (ip))),
bv-bit (bv-vec (bv-vec (bv-vec (ip)))),
bv-bit (ik),
bv-bit (bv-vec (ik)),
bv-bit (bv-vec (bv-vec (ik))),
bv-bit (bv-vec (bv-vec (bv-vec (ik)))),
bvco-carry (cbyp-col (bv-vec (bv-vec (bv-vec (b
bv-vec (bv-vec (bv-vec (b
$ccin$,
nb)))),
bvco-bitvec (cbyp-col (bv-vec (bv-vec (bv-vec (bv-vec (ik)))),
bv-vec (bv-vec (bv-vec (bv-vec (ip)))),
$ccin$,
nb))),
bvco-carry (c4byp (bv-bit (ip),
bv-bit (bv-vec (ip)),
bv-bit (bv-vec (bv-vec (ip))),
bv-bit (bv-vec (bv-vec (bv-vec (ip)))),
bv-bit (ik),
bv-bit (bv-vec (ik)),
bv-bit (bv-vec (bv-vec (ik))),
bv-bit (bv-vec (bv-vec (bv-vec (ik)))),
bvco-carry (cbyp-col (bv-vec (bv-vec (bv-vec (bv-vec (ik))))
bv-vec (bv-vec (bv-vec (bv-vec (ip))))
$ccin$,
nb)))),
bvco-carry (cbyp-col (bv-vec (ik),
bv-vec (ip),

ccin,
nb)))
else bitvec-carry-ovf (bitvec (bvco-carry (cbyp-col (bv-vec (*ik*),
bv-vec (*ip*),
ccin,
nb)),
bvco-bitvec (cbyp-col (bv-vec (*ik*),
bv-vec (*ip*),
ccin,
nb))),
if evenp (bv-size (*ik*))
then alucev-fc (bv-bit (*ik*),
bv-bit (*ip*),
bvco-carry (cbyp-col (bv-vec (*ik*),
bv-vec (*ip*),
ccin,
nb)))
else alucod-fc (bv-bit (*ik*),
bv-bit (*ip*),
bvco-carry (cbyp-col (bv-vec (*ik*),
bv-vec (*ip*),
ccin,
nb))) **endif**,
bvco-carry (cbyp-col (bv-vec (*ik*),
bv-vec (*ip*),
ccin,
nb))) **endif**
else BVCO-NIL **endif**

THEOREM: cbyp-carry2-relate
cbyp-col (*ik*, *ip*, *ccin*, *nb*) = carry-col2 (*ik*, *ip*, *ccin*, *nb*)

THEOREM: cbyp-carry-relate
evenp (*nb*) $\rightarrow$ (cbyp-col (*k*, *p*, *cin*, *nb*) = carry-col (*k*, *p*, *cin*))

EVENT: Disable cbyp-carry2-relate.

EVENT: Disable carry2-carry-relate.

DEFINITION:
alugen-o (*ct3*, *ct2*, *ct1*, *ct0*, *ia*, *ib*)
= **if** falsep (*ia*)
then **if** falsep (*ib*) **then** $\neg$ *ct2*
else $\neg$ *ct3* **endif**

elseif falsep (ib) **then** ¬ $ct1$
else ¬ $ct0$ **endif**

DEFINITION:
alugenod-o ($ct3$, $ct2$, $ct1$, $ct0$, ia, ib)
= **if** falsep (ia)
then if falsep (ib) **then** ¬ $ct3$
else ¬ $ct2$ **endif**
elseif falsep (ib) **then** ¬ $ct0$
else ¬ $ct1$ **endif**

THEOREM: alugen-alugenod-relate
alugen-o ($ct3$, $ct2$, $ct1$, $ct0$, a, b) = alugenod-o ($ct3$, $ct2$, $ct1$, $ct0$, a, ¬ b)

THEOREM: alugenodo-func
(boolp (a) ∧ boolp (b))
→ ((alugenod-o (**t**, **t**, **t**, **t**, a, b) = **f**)
∧ (alugenod-o (**t**, **t**, **t**, **f**, a, b) = (a ∧ (¬ b)))
∧ (alugenod-o (**t**, **t**, **f**, **t**, a, b) = (a ∧ b))
∧ (alugenod-o (**t**, **t**, **f**, **f**, a, b) = a)
∧ (alugenod-o (**t**, **f**, **t**, **t**, a, b) = ((¬ a) ∧ b))
∧ (alugenod-o (**t**, **f**, **t**, **f**, a, b) = exor (a, b))
∧ (alugenod-o (**t**, **f**, **f**, **t**, a, b) = b)
∧ (alugenod-o (**t**, **f**, **f**, **f**, a, b) = (a ∨ b))
∧ (alugenod-o (**f**, **t**, **t**, **t**, a, b) = (¬ (a ∨ b)))
∧ (alugenod-o (**f**, **t**, **t**, **f**, a, b) = (¬ b))
∧ (alugenod-o (**f**, **t**, **f**, **t**, a, b) = (¬ exor (a, b)))
∧ (alugenod-o (**f**, **t**, **f**, **f**, a, b) = (a ∨ (¬ b)))
∧ (alugenod-o (**f**, **f**, **t**, **t**, a, b) = (¬ a))
∧ (alugenod-o (**f**, **f**, **t**, **f**, a, b) = (¬ (a ∧ b)))
∧ (alugenod-o (**f**, **f**, **f**, **t**, a, b) = ((¬ a) ∨ b))
∧ (alugenod-o (**f**, **f**, **f**, **f**, a, b) = **t**))

DEFINITION: cscbi11 ($input$) = (¬ $input$)

DEFINITION: cscbo11 ($input$) = (¬ $input$)

DEFINITION:
prop-col (ina, inb, cp)
= **if** bitvecp (ina)
∧ bitvecp (inb)
∧ controlep (cp)
∧ (bv-size (ina) = bv-size (inb))
then if ina = BV-NIL **then** BV-NIL
else bitvec (alugen-o (cscbi11 (bv-3 (cp)),

cscbi11 (bv-2 (cp)),
cscbi11 (bv-1 (cp)),
cscbi11 (bv-0 (cp)),
bv-bit (ina),
bv-bit (inb)),
prop-col (bv-vec (ina), bv-vec (inb), cp)) **endif**
else BV-NIL **endif**

THEOREM: size-prop-col
(controlep (cp) $\land$ bitvecp (a) $\land$ bitvecp (b) $\land$ (bv-size (a) = bv-size (b)))
$\rightarrow$ (bv-size (prop-col (a, b, cp)) = bv-size (a))

THEOREM: prop-col-5
(bitvecp (a) $\land$ bitvecp (b) $\land$ (bv-size (a) = bv-size (b)))
$\rightarrow$ (prop-col (a, b, nat-to-bv (5, 4)) = bv-not (bv-exor (a, b)))

THEOREM: prop-col-10
(bitvecp (a) $\land$ bitvecp (b) $\land$ (bv-size (a) = bv-size (b)))
$\rightarrow$ (prop-col (a, b, nat-to-bv (10, 4)) = bv-exor (a, b))

THEOREM: prop-col-12
(bitvecp (a) $\land$ bitvecp (b) $\land$ (bv-size (a) = bv-size (b)))
$\rightarrow$ (prop-col (a, b, nat-to-bv (12, 4)) = bv-not (a))

DEFINITION:
kill-col (ina, inb, ck)
= **if** bitvecp (ina)
$\land$ bitvecp (inb)
$\land$ controlep (ck)
$\land$ (bv-size (ina) = bv-size (inb))
then if ina = BV-NIL **then** BV-NIL
else bitvec (alugen-o (cscbi11 (bv-3 (ck)),
cscbi11 (bv-2 (ck)),
cscbi11 (bv-1 (ck)),
cscbi11 (bv-0 (ck)),
bv-bit (ina),
bv-bit (inb)),
kill-col (bv-vec (ina), bv-vec (inb), ck)) **endif**
else BV-NIL **endif**

THEOREM: prop-kill-relate
kill-col (a, b, $control$) = prop-col (a, b, $control$)

THEOREM: size-kill-col
(controlep (ck) $\land$ bitvecp (a) $\land$ bitvecp (b) $\land$ (bv-size (a) = bv-size (b)))
$\rightarrow$ (bv-size (kill-col (a, b, ck)) = bv-size (a))

THEOREM: kill-col-12
(bitvecp (a) ∧ bitvecp (b) ∧ (bv-size (a) = bv-size (b)))
→ (kill-col $(a,\ b,$ nat-to-bv (12, 4)) = bv-not (a))

EVENT: Disable prop-kill-relate.

DEFINITION:
res-col $(ina,\ inb,\ cr)$
= **if** bitvecp (ina)
  ∧ bitvecp (inb)
  ∧ controlep (cr)
  ∧ (bv-size (ina) = bv-size (inb))
 then if ina = BV-NIL **then** BV-NIL
  else bitvec (**if** evenp (bv-size (ina))
    then alugenod-o (cscbi11 (bv-3 (cr)),
      cscbi11 (bv-2 (cr)),
      cscbi11 (bv-1 (cr)),
      cscbi11 (bv-0 (cr)),
      bv-bit (ina),
      bv-bit (inb))
    else alugen-o (cscbi11 (bv-3 (cr)),
      cscbi11 (bv-2 (cr)),
      cscbi11 (bv-1 (cr)),
      cscbi11 (bv-0 (cr)),
      bv-bit (ina),
      bv-bit (inb)) **endif**,
    res-col (bv-vec (ina), bv-vec (inb), cr)) **endif**
 else BV-NIL **endif**

THEOREM: size-res-col
(controlep (cr) ∧ bitvecp (a) ∧ bitvecp (b) ∧ (bv-size (a) = bv-size (b)))
→ (bv-size (res-col $(a,\ b,\ cr)$) = bv-size (a))

THEOREM: res-col-10
(bitvecp (a) ∧ bitvecp (b) ∧ (bv-size (a) = bv-size (b)))
→ (res-col $(a,\ b,$ nat-to-bv (10, 4)) = bv-exor $(a,$ bv-invert-even (b)))

THEOREM: lemma1
(bitvecp (a)
∧ bitvecp (b)
∧ (bv-size (a) = bv-size (b))
∧ controlep (cp)
∧ controlep (ck)
∧ boolp $(ccin)$)

$\rightarrow$ (bv-size (bvco-bitvec (carry-col (kill-col (a, b, ck), prop-col (a, b, cp), $ccin$)))
    = bv-size (prop-col (a, b, cp)))

DEFINITION:
rec-mcalu-imp (nb, byp, ina, inb, ck, cp, cr, $ccin$)
= **if** evenp (nb)
    $\wedge$ bitvecp (ina)
    $\wedge$ bitvecp (inb)
    $\wedge$ boolp ($ccin$)
    $\wedge$ boolp (byp)
    $\wedge$ (bv-size (ina) = bv-size (inb))
  then if truep (byp)
    then bitvec-carry-ovf (res-col (prop-col (ina, inb, cp),
        bvco-bitvec (cbyp-col (kill-col (ina,
            inb,
            ck),
          prop-col (ina,
            inb,
            cp),
          $ccin$,
          nb)),
        cr),
      bvco-carry (cbyp-col (kill-col (ina, inb, ck),
        prop-col (ina, inb, cp),
        $ccin$,
        nb)),
      bvco-ovf (cbyp-col (kill-col (ina, inb, ck),
        prop-col (ina, inb, cp),
        $ccin$,
        nb)))
    else bitvec-carry-ovf (res-col (prop-col (ina, inb, cp),
        bvco-bitvec (carry-col (kill-col (ina,
            inb,
            ck),
          prop-col (ina,
            inb,
            cp),
          $ccin$)),
        cr),
      bvco-carry (carry-col (kill-col (ina,
          inb,
          ck),
        prop-col (ina,
          inb,

```
                                                cp),
                                          ccin)),
                     bvco-ovf (carry-col (kill-col (ina,
                                                   inb,
                                                   ck),
                                         prop-col (ina,
                                                   inb,
                                                   cp),
                                         ccin))) endif
  else BVCO-NIL endif
```

DEFINITION: csexor $(a,\ b)$ = exor $(a,\ b)$

DEFINITION:

```
mcalu (nb, byp, ina, inb, ck, cp, cr, ccin)
=  if evenp (nb)
     ∧  bitvecp (ina)
     ∧  bitvecp (inb)
     ∧  boolp (ccin)
     ∧  boolp (byp)
     ∧  controlep (ck)
     ∧  controlep (cp)
     ∧  controlep (cr)
     ∧  (bv-size (ina) = bv-size (inb))
   then carry-sign-ovf-bitvec (cscbo11 (bvco-carry (rec-mcalu-imp (nb,
                                                                   byp ∧ (11 < nb),
                                                                   ina,
                                                                   inb,
                                                                   ck,
                                                                   cp,
                                                                   cr,
                                                                   cscbi11 (ccin)))),
                              cscbo11 (bv-bit (bvco-bitvec (rec-mcalu-imp (nb,
                                                                           byp ∧ (11 < nb),
                                                                           ina,
                                                                           inb,
                                                                           ck,
                                                                           cp,
                                                                           cr,
                                                                           cscbi11 (ccin))))),
                              csexor (bvco-carry (rec-mcalu-imp (nb,
                                                                 byp ∧ (11 < nb),
                                                                 ina,
                                                                 inb,
```

ck,
cp,
cr,
cscbi11 ($ccin$))),
bvco-ovf (rec-mcalu-imp (nb,
$byp \wedge$ (11 $< nb$),
ina,
inb,
ck,
cp,
cr,
cscbi11 ($ccin$)))),
bvco-bitvec (rec-mcalu-imp (nb,
$byp \wedge$ (11 $< nb$),
ina,
inb,
ck,
cp,
cr,
cscbi11 ($ccin$))))

else CSOBV-NIL **endif**

DEFINITION:
induct-vec-vec-evenp-f (a, b, c)
= **if** bitvecp (a)
$\wedge$ bitvecp (b)
$\wedge$ (bv-size (a) = bv-size (b))
$\wedge$ boolp (c)
then if a = BV-NIL **then t**
elseif evenp (bv-size (a))
then induct-vec-vec-evenp-f (bv-vec (a), bv-vec (b), c)
else induct-vec-vec-evenp-f (bv-vec (a), bv-vec (b), c) **endif**
else t endif

THEOREM: the-proof
(bitvecp (a) $\wedge$ bitvecp (b) $\wedge$ (bv-size (a) = bv-size (b)) $\wedge$ boolp (cin))
$\rightarrow$ (bvco-carry (carry-col (bv-not (a), bv-exor (a, b), cin))
= **if** evenp (bv-size (a)) **then** bvco-carry (bv-adder (a, b, cin))
else $\neg$ bvco-carry (bv-adder (a, b, cin)) **endif**)

THEOREM: the-proof-part2
(bitvecp (a) $\wedge$ bitvecp (b) $\wedge$ (bv-size (a) = bv-size (b)) $\wedge$ boolp (cin))
$\rightarrow$ (bv-exor (bv-exor (a, b),
bv-invert-even (bvco-bitvec (carry-col (bv-not (a),
bv-exor (a, b),

cin))))

$=$ bvco-bitvec (bv-adder (a, b, cin)))

THEOREM: the-proof-part3
(bitvecp (a) $\land$ bitvecp (b) $\land$ (bv-size (a) $=$ bv-size (b)) $\land$ boolp (cin))
$\rightarrow$ (bvco-ovf (carry-col (bv-not (a), bv-exor (a, b), cin))
$=$ **if** evenp (bv-size (a))
then if a = BV-NIL **then f**
else $\neg$ bvco-ovf (bv-adder (a, b, cin)) **endif**
else bvco-ovf (bv-adder (a, b, cin)) **endif**)

THEOREM: rec-mcalu-imp-bv-adder-relate
(bitvecp (a)
$\land$ bitvecp (b)
$\land$ boolp (cin)
$\land$ boolp (byp)
$\land$ (bv-size (a) $=$ bv-size (b))
$\land$ evenp (nb)
$\land$ (bv-size (a) $=$ nb))
$\rightarrow$ (bvco-bitvec (rec-mcalu-imp (nb,
byp,
a,
b,
nat-to-bv (12, 4),
nat-to-bv (10, 4),
nat-to-bv (10, 4),
cin))
$=$ bvco-bitvec (bv-adder (a, b, cin)))

THEOREM: mcalu-imp-bv-adder-relate
(bitvecp (a)
$\land$ bitvecp (b)
$\land$ boolp (cin)
$\land$ boolp (byp)
$\land$ (bv-size (a) $=$ bv-size (b))
$\land$ evenp (nb)
$\land$ (bv-size (a) $=$ nb))
$\rightarrow$ (csobv-bitvec (mcalu (nb,
byp,
a,
b,
nat-to-bv (12, 4),
nat-to-bv (10, 4),
nat-to-bv (10, 4),
cin))

$= \quad$ bvco-bitvec (bv-adder (a, b, $\neg$ cin)))

THEOREM: boolp-not
boolp ($\neg$ a)

THEOREM: tc-interpretation-of-mcalu-output
(bitvecp (a)
$\wedge$ bitvecp (b)
$\wedge$ (bv-size (a) $=$ bv-size (b))
$\wedge$ boolp (cin)
$\wedge$ boolp (byp)
$\wedge$ evenp (nb)
$\wedge$ (bv-size (a) $=$ nb)
$\wedge$ ($a \neq$ BV-NIL)
$\wedge$ ($b \neq$ BV-NIL))
$\rightarrow$ (tc-to-integer (csobv-bitvec (mcalu (nb,
byp,
a,
b,
nat-to-bv (12, 4),
nat-to-bv (10, 4),
nat-to-bv (10, 4),
cin)))
$=$ **if** tc-in-rangep (add (tc-to-integer (a),
add (tc-to-integer (b), carry ($\neg$ cin))),
bv-size (a))
then add (tc-to-integer (a),
add (tc-to-integer (b), carry ($\neg$ cin)))
elseif negativep (add (tc-to-integer (a),
add (tc-to-integer (b), carry ($\neg$ cin))))
then add (tc-to-integer (a),
add (tc-to-integer (b),
add (carry ($\neg$ cin), twoto (bv-size (a)))))
else add (tc-to-integer (a),
add (tc-to-integer (b),
add (carry ($\neg$ cin), $-$ twoto (bv-size (a))))) **endif**)

THEOREM: bv-to-nat-of-bv-not
bv-to-nat (bv-not (a)) $=$ ((twoto (bv-size (a)) $-$ 1) $-$ bv-to-nat (a))

DEFINITION:
tc-minus (a)
$=$ **if** negativep (a) **then** negative-guts (a)
elseif $a \simeq 0$ **then** 0
else $-$ a **endif**

THEOREM: bit-of-bv-not
$(\text{bitvecp}(a) \land (a \neq \text{BV-NIL})) \rightarrow (\text{bv-bit}(\text{bv-not}(a)) = (\neg\ \text{bv-bit}(a)))$

THEOREM: equal-difference-0
$((x - y) = 0) = (y \not< x)$

THEOREM: top-bit-off-implies-smaller
$(\text{bitvecp}(a) \land (a \neq \text{BV-NIL}) \land (\neg\ \text{bv-bit}(a)))$
$\rightarrow \quad (\text{bv-to-nat}(a) < (\text{twoto}(\text{bv-size}(a)) - 1))$

THEOREM: tc-minus-tc-to-integer
$(\text{bitvecp}(a) \land (a \neq \text{BV-NIL}))$
$\rightarrow \quad (\text{tc-minus}(\text{tc-to-integer}(a))$
$\quad = \quad$ **if** tc-to-integer $(a) = 0$ **then** 0
else add $(1,$ tc-to-integer (bv-not $(a)))$ **endif**)

DEFINITION:
tc-fix (x)
$=$ **if** tcp (x) **then** x
else 0 **endif**

THEOREM: tcp-add
$(\text{tcp}(x) \rightarrow \text{tcp}(\text{add}(x, y))) \land (\text{tcp}(y) \rightarrow \text{tcp}(\text{add}(x, y)))$

THEOREM: add-0
$\text{add}(0, x) = \text{tc-fix}(x)$

THEOREM: add-1-1
$\text{add}(1, \text{add}(-1, x)) = \text{tc-fix}(x)$

THEOREM: tc-to-integer-0
$(\text{bitvecp}(a) \land (a \neq \text{BV-NIL}) \land (\text{tc-to-integer}(a) = 0))$
$\rightarrow \quad (\text{tc-to-integer}(\text{bv-not}(a)) = -1)$

THEOREM: bv-not-bv-exor-right
$\text{bv-not}(\text{bv-exor}(a, b)) = \text{bv-exor}(a, \text{bv-not}(b))$

THEOREM: bv-not-nil
$(\text{bitvecp}(a) \land (a \neq \text{BV-NIL})) \rightarrow (\text{bv-not}(a) \neq \text{BV-NIL})$

THEOREM: tc-to-integer-bv-not
$(\text{bitvecp}(b) \land (b \neq \text{BV-NIL}))$
$\rightarrow \quad (\text{tc-to-integer}(\text{bv-not}(b)) = \text{add}(\text{tc-minus}(\text{tc-to-integer}(b)), -1))$

THEOREM: carry-not
$\text{carry}(\neg\ a) = \text{add}(1, \text{tc-minus}(\text{carry}(a)))$

THEOREM: tcp-tc-minus
tcp (tc-minus (a))

THEOREM: mcalu-12-5-10-mcalu-12-10-10-relate
(bitvecp (a)
 $\land$ bitvecp (b)
 $\land$ (bv-size (a) $=$ bv-size (b))
 $\land$ evenp (nb)
 $\land$ boolp (byp)
 $\land$ boolp (cin))
$\rightarrow$ (mcalu (nb,
    byp,
    a,
    b,
    nat-to-bv (12, 4),
    nat-to-bv (5, 4),
    nat-to-bv (10, 4),
    cin)
  $=$ mcalu (nb,
    byp,
    a,
    bv-not (b),
    nat-to-bv (12, 4),
    nat-to-bv (10, 4),
    nat-to-bv (10, 4),
    cin))

THEOREM: tcp-twoto
tcp (twoto (a))

THEOREM: tcp-minus-twoto
($a \in \mathbf{N}$) $\rightarrow$ tcp ($-$ twoto (a))

THEOREM: tc-interpretation-of-mcalu-imp-add
(bitvecp (a)
 $\land$ bitvecp (b)
 $\land$ (bv-size (a) $=$ bv-size (b))
 $\land$ ($a \neq$ BV-NIL)
 $\land$ ($b \neq$ BV-NIL)
 $\land$ boolp (byp)
 $\land$ evenp (nb)
 $\land$ (bv-size (a) $=$ nb))
$\rightarrow$ (tc-to-integer (csobv-bitvec (mcalu (nb,
    byp,
    a,

b,
nat-to-bv (12, 4),
nat-to-bv (10, 4),
nat-to-bv (10, 4),
t)))

= **if** tc-in-rangep (add (tc-to-integer (a), tc-to-integer (b)),
bv-size (a))
then add (tc-to-integer (a), tc-to-integer (b))
elseif negativep (add (tc-to-integer (a), tc-to-integer (b)))
then add (tc-to-integer (a),
add (tc-to-integer (b), twoto (bv-size (a))))
else add (tc-to-integer (a),
add (tc-to-integer (b), − twoto (bv-size (a)))) **endif**)

THEOREM: tc-interpretation-of-mcalu-imp-xsub
(bitvecp (a)
∧ bitvecp (b)
∧ (bv-size (a) = bv-size (b))
∧ ($a \neq$ BV-NIL)
∧ ($b \neq$ BV-NIL)
∧ boolp (cin)
∧ boolp (byp)
∧ evenp (nb)
∧ (bv-size (a) = nb))
→ (tc-to-integer (csobv-bitvec (mcalu (nb,
byp,
a,
b,
nat-to-bv (12, 4),
nat-to-bv (5, 4),
nat-to-bv (10, 4),
cin)))

= **if** tc-in-rangep (add (tc-to-integer (a),
add (tc-minus (tc-to-integer (b)),
tc-minus (carry (cin)))),
bv-size (a))
then add (tc-to-integer (a),
add (tc-minus (tc-to-integer (b)), tc-minus (carry (cin))))
elseif negativep (add (tc-to-integer (a),
add (tc-minus (tc-to-integer (b)),
tc-minus (carry (cin)))))
then add (tc-to-integer (a),
add (tc-minus (tc-to-integer (b)),
add (tc-minus (carry (cin)), twoto (bv-size (a)))))

else add (tc-to-integer (a),
    add (tc-minus (tc-to-integer (b)),
        add (tc-minus (carry (cin)),
            $-$ twoto (bv-size (a))))) **endif**)

THEOREM: tc-interpretation-of-mcalu-imp-sub
(bitvecp (a)
 $\land$ bitvecp (b)
 $\land$ (bv-size (a) $=$ bv-size (b))
 $\land$ ($a \neq$ BV-NIL)
 $\land$ ($b \neq$ BV-NIL)
 $\land$ boolp (byp)
 $\land$ evenp (nb)
 $\land$ (bv-size (a) $=$ nb))
 $\rightarrow$ (tc-to-integer (csobv-bitvec (mcalu (nb,
    byp,
    a,
    b,
    nat-to-bv (12, 4),
    nat-to-bv (5, 4),
    nat-to-bv (10, 4),
    f)))
 $=$ **if** tc-in-rangep (add (tc-to-integer (a),
        tc-minus (tc-to-integer (b))),
      bv-size (a))
  then add (tc-to-integer (a), tc-minus (tc-to-integer (b)))
  elseif negativep (add (tc-to-integer (a),
        tc-minus (tc-to-integer (b))))
  then add (tc-to-integer (a),
      add (tc-minus (tc-to-integer (b)), twoto (bv-size (a))))
  else add (tc-to-integer (a),
      add (tc-minus (tc-to-integer (b)),
        $-$ twoto (bv-size (a)))) **endif**)

THEOREM: bv-adder-non-nil2
(bitvecp (a)
 $\land$ bitvecp (b)
 $\land$ (bv-size (a) $=$ bv-size (b))
 $\land$ boolp (cin)
 $\land$ ($a \neq$ BV-NIL))
 $\rightarrow$ ((bvco-bitvec (bv-adder (a, b, cin)) $=$ BV-NIL) $=$ **f**)

DEFINITION:
last (it)
$=$ **if** bitvecp (it) $\land$ ($it \neq$ BV-NIL)

then if bv-vec (it) = BV-NIL **then** bv-bit (it)
else last (bv-vec (it)) **endif**
else f endif

DEFINITION:
butlast (it)
= **if** bitvecp (it) $\land$ ($it \neq$ BV-NIL)
then if bv-vec (it) = BV-NIL **then** BV-NIL
else bitvec (bv-bit (it), butlast (bv-vec (it))) **endif**
else BV-NIL **endif**

DEFINITION:
do-shift-1 (it, *shiftin*)
= **if** bitvecp (it) $\land$ ($it \neq$ BV-NIL) $\land$ boolp (*shiftin*)
then bitvec (*shiftin*, butlast (it))
else BV-NIL **endif**

DEFINITION:
up-shift-1 (it, *shiftin*)
= **if** bitvecp (it) $\land$ ($it \neq$ BV-NIL) $\land$ boolp (*shiftin*)
then bv-append (bv-vec (it), bitvec (*shiftin*, BV-NIL))
else BV-NIL **endif**

DEFINITION:
do-shift-n (n, it, *shiftin*)
= **if** ($n \in \mathbf{N}$) $\land$ bitvecp (it) $\land$ boolp (*shiftin*)
then if $n \simeq 0$ **then** it
else do-shift-1 (do-shift-n ($n - 1$, it, *shiftin*), *shiftin*) **endif**
else BV-NIL **endif**

DEFINITION:
up-shift-n (n, it, *shiftin*)
= **if** ($n \in \mathbf{N}$) $\land$ bitvecp (it) $\land$ boolp (*shiftin*)
then if $n \simeq 0$ **then** it
else up-shift-1 (up-shift-n ($n - 1$, it, *shiftin*), *shiftin*) **endif**
else BV-NIL **endif**

THEOREM: size-bv-append
(bitvecp (a) $\land$ bitvecp (b))
$\rightarrow$ (bv-size (bv-append (a, b)) = (bv-size (a) + bv-size (b)))

THEOREM: append-not-nil
(bitvecp (a) $\land$ bitvecp (b) $\land$ ($a \neq$ BV-NIL))
$\rightarrow$ ((bv-append (a, b) = BV-NIL) = **f**)

THEOREM: vec-append
$((a \neq \text{BV-NIL}) \wedge \text{bitvecp}(a) \wedge \text{bitvecp}(b))$
$\rightarrow \quad (\text{bv-vec}(\text{bv-append}(a, b)) = \text{bv-append}(\text{bv-vec}(a), b))$

THEOREM: size-up-shift-1
$\text{boolp}(\mathit{shiftin}) \rightarrow (\text{bv-size}(\text{up-shift-1}(\mathit{it}, \mathit{shiftin})) = \text{bv-size}(\mathit{it}))$

THEOREM: size-up-shift-n
$(\text{boolp}(\mathit{shiftin}) \wedge (n \in \mathbf{N}))$
$\rightarrow \quad (\text{bv-size}(\text{up-shift-n}(n, \mathit{it}, \mathit{shiftin})) = \text{bv-size}(\mathit{it}))$

THEOREM: size-do-shift-n
$(\text{boolp}(\mathit{shiftin}) \wedge (n \in \mathbf{N}))$
$\rightarrow \quad (\text{bv-size}(\text{do-shift-n}(n, \mathit{it}, \mathit{shiftin})) = \text{bv-size}(\mathit{it}))$

THEOREM: bv-append-not-not
$(\text{bitvecp}(a) \wedge \text{bitvecp}(b))$
$\rightarrow \quad (\text{bv-not}(\text{bv-append}(a, b)) = \text{bv-append}(\text{bv-not}(a), \text{bv-not}(b)))$

THEOREM: up-shift-0
$(\text{bitvecp}(\mathit{it}) \wedge \text{boolp}(\mathit{sin})) \rightarrow (\text{up-shift-n}(\mathtt{0}, \mathit{it}, \mathit{sin}) = \mathit{it})$

THEOREM: hack2
$((b \in \mathbf{N}) \wedge (a \in \mathbf{N})) \rightarrow ((a + (\mathtt{0} * b)) = a)$

THEOREM: evenp-twoto
$(a \not\simeq \mathtt{0}) \rightarrow \text{evenp}(\text{twoto}(a))$

THEOREM: evenp-plus
$((a \in \mathbf{N}) \wedge (b \in \mathbf{N}) \wedge \text{evenp}(a) \wedge \text{evenp}(b)) \rightarrow \text{evenp}(a + b)$

THEOREM: evenp-plus-extended
$((a \in \mathbf{N}) \wedge (b \in \mathbf{N}))$
$\rightarrow \quad (\text{evenp}(a + b)$
$\qquad = \quad \textbf{if } \text{evenp}(a) \ \textbf{then } \text{evenp}(b)$
$\qquad\qquad \textbf{else } \neg \text{evenp}(b) \ \textbf{endif})$

THEOREM: do-shift-strict
$\text{do-shift-n}(n, \text{BV-NIL}, \mathit{sin}) = \text{BV-NIL}$

THEOREM: do-shift-0
$(\text{bitvecp}(\mathit{it}) \wedge \text{boolp}(\mathit{sin})) \rightarrow (\text{do-shift-n}(\mathtt{0}, \mathit{it}, \mathit{sin}) = \mathit{it})$

DEFINITION:
$\text{mult}(a, b)$
$= \quad \textbf{if } \text{negativep}(a)$
$\quad \textbf{then if } \text{negativep}(b) \ \textbf{then } \text{negative-guts}(a) * \text{negative-guts}(b)$
$\qquad\quad \textbf{else } \text{tc-minus}(\text{negative-guts}(a) * b) \ \textbf{endif}$
$\quad \textbf{elseif } \text{negativep}(b) \ \textbf{then } \text{tc-minus}(a * \text{negative-guts}(b))$
$\quad \textbf{else } a * b \ \textbf{endif}$

THEOREM: tcp-mult
tcp (mult $(a,\ b)$)

THEOREM: commutativity2-of-mult
mult $(x,$ mult $(y,\ z))$ = mult $(y,$ mult $(x,\ z))$

THEOREM: commutativity-of-mult
mult $(x,\ y)$ = mult $(y,\ x)$

THEOREM: associativity-of-mult
mult (mult $(x,\ y),\ z)$ = mult $(x,$ mult $(y,\ z))$

THEOREM: bv-to-nat-bv-append
(bitvecp (a) $\land$ bitvecp (b))
$\rightarrow$ (bv-to-nat (bv-append $(a,\ b)$)
= ((bv-to-nat (a) $*$ twoto (bv-size (b))) + bv-to-nat (b)))

THEOREM: bv-to-nat-of-vec
bv-to-nat (bv-vec (a))
= **if** bv-bit (a) **then** add (bv-to-nat (a), $-$ twoto (bv-size (bv-vec (a))))
else bv-to-nat (a) **endif**

THEOREM: times-difference
$(a * (x - y)) = ((a * x) - (a * y))$

THEOREM: not-bit-implies-in-range-times-2
($\neg$ bv-bit (a)) $\rightarrow$ ((2 $*$ bv-to-nat (a)) $<$ twoto (bv-size (a)))

THEOREM: remainder-natural-interpretation-of-up-shift-1
bv-to-nat (up-shift-1 $(it,$ **f**))
= ((bv-to-nat (it) $*$ 2) **mod** twoto (bv-size (it)))

THEOREM: remainder-remainder
$((e$ **mod** $b)$ **mod** $b) = (e$ **mod** $b)$

THEOREM: remainder-diff
$(a \not< b) \rightarrow (((a - b)$ **mod** $b) = (a$ **mod** $b))$

THEOREM: remainder-diff-times
$(x \not< (n * b)) \rightarrow (((x - (n * b))$ **mod** $b) = (x$ **mod** $b))$

DEFINITION:
abs (a)
= **if** $a \in \mathbf{N}$ **then** a
elseif negativep (a) **then** negative-guts (a)
else 0 **endif**

DEFINITION:
divide $(a,\ b)$
$=$ **if** negativep (a)
then if negativep (b) **then** negative-guts (a) $\div$ negative-guts (b)
else tc-minus (negative-guts (a) $\div$ b) **endif**
elseif negativep (b) **then** tc-minus (a $\div$ negative-guts (b))
else $a \div b$ **endif**

THEOREM: tcp-divide
tcp (divide $(a,\ b)$)

THEOREM: hackxaux
$((w \in \mathbf{N})$
$\wedge$ $(w < b)$
$\wedge$ $(z \in \mathbf{N})$
$\wedge$ $(z < c)$
$\wedge$ $(c \in \mathbf{N})$
$\wedge$ $(c \neq 0)$
$\wedge$ $(b \neq 0)$
$\wedge$ $(b \in \mathbf{N}))$
$\rightarrow$ $((z + (c * w)) < (b * c))$

THEOREM: times-to-mult-1
(tcp (x) $\wedge$ tc-in-rangep $(x,\ z)$)
$\rightarrow$ (nat-to-integer ((2 $*$ integer-to-nat $(x,\ z)$) **mod** twoto (z), z)
$=$ **if** tc-in-rangep (mult (2, x), z) **then** mult (2, x)
elseif negativep (mult (2, x)) **then** add (mult (2, x), twoto (z))
else add (mult (2, x), $-$ twoto (z)) **endif**)

THEOREM: integer-interpretation-of-up-shift-1
(bitvecp (it) $\wedge$ $(it \neq$ BV-NIL$)$)
$\rightarrow$ (tc-to-integer (up-shift-1 (it, **f**))
$=$ **if** tc-in-rangep (mult (tc-to-integer (it), 2), bv-size (it))
then mult (tc-to-integer (it), 2)
elseif negativep (mult (tc-to-integer (it), 2))
then add (mult (tc-to-integer (it), 2), twoto (bv-size (it)))
else add (mult (tc-to-integer (it), 2),
$-$ twoto (bv-size (it))) **endif**)

THEOREM: lessp-plus-1
$((a + c) < (b + c)) = (a < b)$

THEOREM: lessp-plus-1-commuted
$((a + c) < (c + b)) = (a < b)$

Theorem: lessp-plus-2
$(b < c) \rightarrow ((a < (c - b)) = ((a + b) < c))$

Theorem: equal-diff-twoto
$(\text{twoto}\,(a) - (\text{twoto}\,(a - 1) + b)) = ((\text{twoto}\,(a) \div 2) - b)$

Theorem: evenp-times-even
$(\text{evenp}\,(a) \vee \text{evenp}\,(b)) \rightarrow \text{evenp}\,(a * b)$

Theorem: real-hack-1
$((z < 2) \wedge \text{evenp}\,(z + (2 * a)))$
$\rightarrow \quad ((z + (2 * a)) = (2 * a))$

Theorem: real-hack-6
$((w < 2) \wedge (\neg\, \text{evenp}\,(w))) \rightarrow ((w - 1) = 0)$

Theorem: not-even-add1-commuted
$((a \in \mathbf{N}) \wedge (\neg\, \text{evenp}\,(1 + a))) \rightarrow \text{evenp}\,(a)$

Theorem: evenp-add1-commuted
$\text{evenp}\,(1 + a) \rightarrow (\neg\, \text{evenp}\,(a))$

Definition: $\text{multiplep}\,(a, n) = ((a \;\mathbf{mod}\; n) \simeq 0)$

Theorem: negative-guts-tc-minus
$(a \in \mathbf{N}) \rightarrow (\text{negative-guts}\,(\text{tc-minus}\,(a)) = a)$

Definition:
$\text{invert-lsb}\,(it) = \text{bv-append}\,(\text{butlast}\,(it), \text{bitvec}\,(\neg\, \text{last}\,(it), \textsc{bv-nil}))$

Theorem: lsb-implies-odd
$\text{last}\,(it) = (\neg\, \text{evenp}\,(\text{bv-to-nat}\,(it)))$

Theorem: bv-bit-of-bv-append
$(\text{bitvecp}\,(a) \wedge (a \neq \textsc{bv-nil}) \wedge \text{bitvecp}\,(b))$
$\rightarrow \quad (\text{bv-bit}\,(\text{bv-append}\,(a, b)) = \text{bv-bit}\,(a))$

Theorem: butlast-bv-append
$(\text{bitvecp}\,(b) \wedge (b \neq \textsc{bv-nil}))$
$\rightarrow \quad (\text{butlast}\,(\text{bv-append}\,(a, b)) = \text{bv-append}\,(a, \text{butlast}\,(b)))$

Theorem: bv-append-of-bv-nil
$\text{bitvecp}\,(a) \rightarrow (\text{bv-append}\,(a, \textsc{bv-nil}) = a)$

Theorem: up-shift-1-invert-lsb
$(\text{bitvecp}\,(it) \wedge (it \neq \textsc{bv-nil}))$
$\rightarrow \quad (\text{up-shift-1}\,(it, \mathbf{t}) = \text{invert-lsb}\,(\text{up-shift-1}\,(it, \mathbf{f})))$

THEOREM: last-of-bv-append
(bitvecp (a) $\wedge$ bitvecp (b) $\wedge$ $(b \neq$ BV-NIL$))$
$\rightarrow$ (last (bv-append $(a,\ b)$) = last (b))

THEOREM: quotient-diff-times
$((x - (i * j)) \div j)$
$=$ **if** $j \simeq 0$ **then** 0
else $(x \div j) - i$ **endif**

THEOREM: times-quotient-lessp-relate
$(a < (b * c)) \rightarrow ((a \div b) < c)$

DEFINITION:
induct-quot-quot $(x,\ y,\ b)$
$=$ **if** $b \simeq 0$ **then t**
elseif $x < b$ **then t**
else induct-quot-quot $(x - b,\ y - b,\ b)$ **endif**

THEOREM: quotient-lessp
$((x \in \mathbf{N}) \wedge (y \in \mathbf{N}) \wedge (x \not< y)) \rightarrow ((x \div b) \not< (y \div b))$

THEOREM: times-quotient-lessp-relate-dual
$((a \in \mathbf{N}) \wedge (b \in \mathbf{N}) \wedge (c \in \mathbf{N}) \wedge (b \neq 0) \wedge (a \not< (b * c)))$
$\rightarrow$ $((a \div b) \not< c)$

DEFINITION:
new-r $(i,\ r,\ noem)$
$=$ **if** $(noem \in \mathbf{N}) = (r \in \mathbf{N})$
then add (r, tc-minus (mult ($noem$, twoto $(i - 1)$)))
else add (r, mult ($noem$, twoto $(i - 1)$)) **endif**

DEFINITION:
anrd $(i,\ r,\ noem)$
$=$ **if** $i \simeq 0$ **then** BV-NIL
else bitvec (($noem \in \mathbf{N}$) = (new-r $(i,\ r,\ noem) \in \mathbf{N}$),
anrd ($i - 1$, new-r $(i,\ r,\ noem)$, $noem$)) **endif**

THEOREM: tcp-new-r
tcp (new-r $(i,\ r,\ noem)$)

THEOREM: size-anrd-bv
bv-size (anrd $(i,\ r,\ noem)$) = fix (i)

THEOREM: anrd-i-0
$(i \simeq 0) \rightarrow$ (tc-to-integer (anrd $(i,\ tel,\ noem)$) = 0)

THEOREM: quotient-times-lessp
$((a \in \mathbf{N}) \wedge (b \in \mathbf{N}) \wedge (c \in \mathbf{N}) \wedge (b \neq 0))$
$\rightarrow \quad ((a < (b * c)) = ((a \div b) < c))$

THEOREM: quotient-times-commuted
$((j * i) \div i)$
$= \quad$ **if** $i \simeq 0$ **then** 0
else fix (j) **endif**

THEOREM: multiplep-diff
$((a \not< j) \wedge \text{multiplep}(a, j)) \rightarrow \text{multiplep}(a - j, j)$

THEOREM: q-d-t-lemma1
$((a \in \mathbf{N}) \wedge (a \neq 0))$
$\rightarrow \quad (((a - 1) \div j)$
$= \quad$ **if** multiplep (a, j) **then** $(a \div j) - 1$
else $a \div j$ **endif**)

THEOREM: q-d-t-lemma2
$((i \in \mathbf{N}) \wedge (j \in \mathbf{N}) \wedge (j \neq 0))$
$\rightarrow \quad ((((i * j) - 1) \div j) = (i - 1))$

THEOREM: q-d-t-lemma3
$((j \in \mathbf{N}) \wedge (j \neq 0) \wedge (x \in \mathbf{N}) \wedge (x \neq 0) \wedge (x < j))$
$\rightarrow \quad ((((i * j) - x) \div j) = (i - 1))$

THEOREM: quotient-diff-times-commuted
$(((i * j) - x) \div j)$
$= \quad$ **if** $j \simeq 0$ **then** 0
elseif multiplep (x, j) **then** $i - (x \div j)$
else $(i - (x \div j)) - 1$ **endif**

DEFINITION:
imultiplep (a, b)
$= \quad$ **if** $a \in \mathbf{N}$
then if $b \in \mathbf{N}$ **then** multiplep (a, b)
else multiplep $(a$, negative-guts $(b))$ **endif**
elseif $b \in \mathbf{N}$ **then** multiplep (negative-guts $(a), b)$
else multiplep (negative-guts (a), negative-guts $(b))$ **endif**

THEOREM: imultiplep-add
(tcp $(a) \wedge$ tcp $(b) \wedge (x \in \mathbf{N}) \wedge$ imultiplep $(a, b))$
$\rightarrow \quad$ imultiplep (add $(a$, mult $(b$, twoto $(x))), b)$

THEOREM: tc-minus-mult
tc-minus (mult $(a, b))$ = mult (tc-minus $(a), b)$

THEOREM: multiplep-tc-minus
imultiplep $(a,\ \text{tc-minus}(b)) = \text{imultiplep}(a,\ b)$

THEOREM: imultiplep-new-r
$(\text{tcp}(r) \wedge \text{tcp}(n) \wedge (i \in \mathbf{N}) \wedge \text{imultiplep}(r,\ n))$
$\rightarrow \quad \text{imultiplep}(\text{new-r}(i,\ r,\ n),\ n)$

THEOREM: remainder-diff-times-commuted
$((a \in \mathbf{N}) \wedge (x \in \mathbf{N}) \wedge (b \in \mathbf{N}) \wedge (x < b) \wedge ((a * b) \not< x))$
$\rightarrow \quad ((((a * b) - x) \ \mathbf{mod}\ b)$
$\quad = \quad \mathbf{if}\ x = 0\ \mathbf{then}\ 0$
$\quad\quad \mathbf{else}\ b - x\ \mathbf{endif})$

DEFINITION:
times-times-induct $(a,\ b,\ c)$
$= \quad \mathbf{if}\ b \simeq 0\ \mathbf{then}\ \mathbf{t}$
$\quad \mathbf{else}\ \text{times-times-induct}(a,\ b - 1,\ c - 1)\ \mathbf{endif}$

THEOREM: not-equal-times-0
$((a \in \mathbf{N}) \wedge (b \in \mathbf{N}) \wedge (a \neq 0) \wedge (b \neq 0))$
$\rightarrow \quad ((0 * a) \neq (a * b))$

THEOREM: equal-times-times
$((a \in \mathbf{N}) \wedge (b \in \mathbf{N}) \wedge (c \in \mathbf{N}) \wedge (a \neq 0))$
$\rightarrow \quad (((a * b) = (a * c)) = (b = c))$

THEOREM: hack-around-next-lemma
$((w \in \mathbf{N})$
$\wedge \quad (z \in \mathbf{N})$
$\wedge \quad (z < b)$
$\wedge \quad (v \in \mathbf{N})$
$\wedge \quad (b \neq 0)$
$\wedge \quad (b \in \mathbf{N})$
$\wedge \quad (w \neq 0)$
$\wedge \quad (z \neq 0)$
$\wedge \quad ((b + (b * w)) \not< (z + (b * v)))$
$\wedge \quad ((b * w) < (z + (b * v))))$
$\rightarrow \quad (((1 + v) - 1) = w)$

THEOREM: lower-upper-determines
$((a \in \mathbf{N})$
$\wedge \quad (b \in \mathbf{N})$
$\wedge \quad (a \neq 0)$
$\wedge \quad (a \neq 1)$
$\wedge \quad (x \in \mathbf{N})$

$\wedge \quad (x \neq 0)$
$\wedge \quad ((x \ \mathbf{mod}\ b) \neq 0)$
$\wedge \quad ((a * b) \not< x)$
$\wedge \quad (((a - 1) * b) < x))$
$\rightarrow \quad ((1 + (x \div b)) = a)$

THEOREM: remainder-diff-times-commuted-2
$((a \in \mathbf{N}) \wedge (x \in \mathbf{N}) \wedge (b \in \mathbf{N}) \wedge (x \not< b) \wedge ((a * b) \not< x))$
$\rightarrow \quad ((((a * b) - x) \ \mathbf{mod}\ b)$
$\quad = \ \mathbf{if}\ (x \ \mathbf{mod}\ b) = 0 \ \mathbf{then}\ 0$
$\quad\quad \mathbf{else}\ b - (x \ \mathbf{mod}\ b) \ \mathbf{endif})$

THEOREM: not-imultiplep-add
$(\text{tcp}(a) \wedge \text{tcp}(b) \wedge (x \in \mathbf{N}) \wedge (\neg \text{ imultiplep}(a, b)))$
$\rightarrow \quad (\neg \text{ imultiplep}(\text{add}(a, \text{mult}(b, \text{twoto}(x))), b))$

THEOREM: not-imultiplep-new-r
$(\text{tcp}(n) \wedge \text{tcp}(r) \wedge (i \in \mathbf{N}) \wedge (\neg \text{ imultiplep}(r, n)))$
$\rightarrow \quad (\neg \text{ imultiplep}(\text{new-r}(i, r, n), n))$

THEOREM: unfold-positive-tc-to-integer
$((\text{tc-to-integer}(a) = b) \wedge (b \in \mathbf{N}))$
$\rightarrow \quad (\text{tc-to-integer}(a) = \text{bv-to-nat}(a))$

THEOREM: unfold-negative-tc-to-integer
$((\text{tc-to-integer}(a) = b) \wedge \text{negativep}(b))$
$\rightarrow \quad (\text{tc-to-integer}(a) = \text{add}(\text{tc-minus}(\text{twoto}(\text{bv-size}(a))), \text{bv-to-nat}(a)))$

DEFINITION:
$\text{div-in-rangep}(tel, noem, i)$
$= \quad \mathbf{if}\ tel \in \mathbf{N}$
$\quad \mathbf{then}\ \mathbf{if}\ noem \in \mathbf{N}\ \mathbf{then}\ tel < (\text{twoto}(i - 1) * noem)$
$\quad\quad \mathbf{else}\ tel < (\text{twoto}(i - 1) * \text{negative-guts}(noem))\ \mathbf{endif}$
$\quad \mathbf{elseif}\ noem \in \mathbf{N}\ \mathbf{then}\ (\text{twoto}(i - 1) * noem) \not< \text{negative-guts}(tel)$
$\quad \mathbf{else}\ (\text{twoto}(i - 1) * \text{negative-guts}(noem))$
$\quad\quad \not< \quad \text{negative-guts}(tel)\ \mathbf{endif}$

THEOREM: div-in-rangep-new-r-sign-1
$(\text{div-in-rangep}(tel, noem, i) \wedge (tel \in \mathbf{N}) \wedge (noem \in \mathbf{N}))$
$\rightarrow \quad \text{negativep}(\text{add}(tel, \text{tc-minus}(\text{mult}(noem, \text{twoto}(i - 1)))))$

THEOREM: div-in-rangep-new-r-sign-2
$(\text{div-in-rangep}(tel, noem, i) \wedge (tel \in \mathbf{N}) \wedge \text{negativep}(noem))$
$\rightarrow \quad \text{negativep}(\text{add}(tel, \text{mult}(noem, \text{twoto}(i - 1))))$

THEOREM: div-in-rangep-new-r-sign-3
$(\text{div-in-rangep}(tel, noem, i) \land \text{negativep}(tel) \land (noem \in \mathbf{N}))$
$\rightarrow \quad (\text{add}(tel, \text{mult}(noem, \text{twoto}(i - 1))) \in \mathbf{N})$

THEOREM: div-in-rangep-new-r-sign-4
$(\text{div-in-rangep}(tel, noem, i) \land \text{negativep}(tel) \land \text{negativep}(noem))$
$\rightarrow \quad (\text{add}(tel, \text{tc-minus}(\text{mult}(noem, \text{twoto}(i - 1)))) \in \mathbf{N})$

THEOREM: quotient-diff-times-dual
$((x - (j * i)) \div j)$
$= \quad$ **if** $j \simeq 0$ **then** 0
else $(x \div j) - i$ **endif**

THEOREM: quotient-diff-times-commuted-dual
$(((j * i) - x) \div j)$
$= \quad$ **if** $j \simeq 0$ **then** 0
elseif multiplep (x, j) **then** $i - (x \div j)$
else $(i - (x \div j)) - 1$ **endif**

THEOREM: sub1-difference
$((a - b) - 1) = ((a - 1) - b)$

THEOREM: quotient-times-lessp-refrased
$((a \in \mathbf{N}) \land (b \in \mathbf{N}) \land (c \in \mathbf{N}) \land (b \neq 0))$
$\rightarrow \quad (((a \div b) < c) = (a < (b * c)))$

EVENT: Disable quotient-times-lessp.

THEOREM: difference-1
$(x - 1) = (x - 1)$

THEOREM: may-be-baby
$((x \in \mathbf{N}) \land (y \in \mathbf{N}) \land (z \in \mathbf{N}) \land (y \neq 0) \land (x < (z * y)))$
$\rightarrow \quad ((z - 1) \not< (x \div y))$

THEOREM: may-be-baby-2
$((x \in \mathbf{N}) \land (y \in \mathbf{N}) \land (z \in \mathbf{N}) \land (y \neq 0) \land ((y * z) \not< x))$
$\rightarrow \quad (z \not< (x \div y))$

THEOREM: may-be-baby-3
$(\text{negativep}(x)$
$\land \quad (\text{negative-guts}(x) \neq 0)$
$\land \quad (y \in \mathbf{N})$
$\land \quad (z \in \mathbf{N})$
$\land \quad (y \neq 0)$

$\wedge$ ((negative-guts (x) **mod** $y) \neq 0$)
$\wedge$ $((y * z) \not< \text{negative-guts}(x))$
$\wedge$ ((negative-guts $(x) \div y) = 0$)
$\wedge$ $(z < 1))$
$\rightarrow$ (divide $((y * z) - \text{negative-guts}(x),\ y) = \text{add}(-\ (1 - z),\ 0))$

THEOREM: may-be-baby-4
$((x \in \mathbf{N})$
$\wedge$ $(y \in \mathbf{N})$
$\wedge$ $(z \in \mathbf{N})$
$\wedge$ $(z \not< 1)$
$\wedge$ $(x \neq 0)$
$\wedge$ $(y \neq 0)$
$\wedge$ $((y * z) \not< x)$
$\wedge$ $((x \ \mathbf{mod}\ y) \neq 0))$
$\rightarrow$ $((z - 1) \not< (x \div y))$

THEOREM: may-be-baby-5
$((x \in \mathbf{N})$
$\wedge$ $(x \neq 0)$
$\wedge$ $(y \in \mathbf{N})$
$\wedge$ $(y \neq 0)$
$\wedge$ $(y \not< x)$
$\wedge$ $((x \ \mathbf{mod}\ y) \neq 0))$
$\rightarrow$ $((x \div y) = 0)$

THEOREM: may-be-baby-6
(negativep (x)
$\wedge$ (negative-guts $(x) \neq 0$)
$\wedge$ negativep (y)
$\wedge$ (negative-guts $(y) \neq 0$)
$\wedge$ negativep (z)
$\wedge$ (negative-guts $(z) \neq 0$)
$\wedge$ ((negative-guts (x) **mod** negative-guts $(y)) \neq 0$)
$\wedge$ ((negative-guts (y) $*$ negative-guts $(z)) \not<$ negative-guts (x))
$\wedge$ ($1 <$ negative-guts (z)))
$\rightarrow$ (divide ((negative-guts (y) $*$ negative-guts (z)) $-$ negative-guts (x),
y)
$=$ add (negative-guts (x) $\div$ negative-guts (y),
$-$ (negative-guts $(z) - 1$)))

THEOREM: equal-times-times-commuted
$((a \in \mathbf{N}) \wedge (b \in \mathbf{N}) \wedge (c \in \mathbf{N}) \wedge (a \neq 0))$
$\rightarrow$ $(((a * b) = (c * a)) = (b = c))$

THEOREM: divide-add-mult
$(\text{tcp}(x) \land \text{tcp}(y) \land \text{tcp}(z))$
$\rightarrow \quad (\text{divide}(\text{add}(x,\ \text{mult}(y,\ z)),\ y)$
$= \quad \textbf{if } y = 0 \textbf{ then } 0$
$\textbf{elseif } \text{imultiplep}(x,\ y) \textbf{ then } \text{add}(\text{divide}(x,\ y),\ z)$
$\textbf{elseif } x \in \mathbf{N}$
$\textbf{then if } \text{add}(x,\ \text{mult}(y,\ z)) \in \mathbf{N} \textbf{ then } \text{add}(\text{divide}(x,\ y),\ z)$
$\textbf{elseif } y \in \mathbf{N} \textbf{ then } \text{add}(\text{divide}(x,\ y),\ \text{add}(z,\ 1))$
$\textbf{else } \text{add}(\text{divide}(x,\ y),\ \text{add}(z,\ -1)) \textbf{ endif}$
$\textbf{elseif } \text{negativep}(\text{add}(x,\ \text{mult}(y,\ z))) \textbf{ then } \text{add}(\text{divide}(x,\ y),\ z)$
$\textbf{elseif } y \in \mathbf{N} \textbf{ then } \text{add}(\text{divide}(x,\ y),\ \text{add}(z,\ -1))$
$\textbf{else } \text{add}(\text{divide}(x,\ y),\ \text{add}(z,\ 1)) \textbf{ endif})$

THEOREM: div-in-range-sign-new-r-1
$((tel \in \mathbf{N}) \land (noem \in \mathbf{N}) \land \text{div-in-rangep}(tel,\ noem,\ i))$
$\rightarrow \quad \text{negativep}(\text{new-r}(i,\ tel,\ noem))$

THEOREM: div-in-range-sign-new-r-2
$((tel \in \mathbf{N}) \land \text{negativep}(noem) \land \text{div-in-rangep}(tel,\ noem,\ i))$
$\rightarrow \quad \text{negativep}(\text{new-r}(i,\ tel,\ noem))$

THEOREM: div-in-range-sign-new-r-3
$(\text{negativep}(tel) \land (noem \in \mathbf{N}) \land \text{div-in-rangep}(tel,\ noem,\ i))$
$\rightarrow \quad (\text{new-r}(i,\ tel,\ noem) \in \mathbf{N})$

THEOREM: div-in-range-sign-new-r-4
$(\text{negativep}(tel) \land \text{negativep}(noem) \land \text{div-in-rangep}(tel,\ noem,\ i))$
$\rightarrow \quad (\text{new-r}(i,\ tel,\ noem) \in \mathbf{N})$

THEOREM: elim-hyp-1a-3aa
$(\text{tcp}(x) \land \text{negativep}(x) \land (y \in \mathbf{N}) \land (y \neq 0) \land \text{imultiplep}(x,\ y))$
$\rightarrow \quad \text{negativep}(\text{divide}(x,\ y))$

THEOREM: elim-hyp-1a-3ab
$(\text{negativep}(x) \land (y \in \mathbf{N})) \rightarrow \text{negativep}(\text{add}(-1,\ \text{divide}(x,\ y)))$

THEOREM: transfer-add
$(\text{tcp}(b) \land \text{tcp}(c)) \rightarrow ((\text{add}(a,\ b) = c) = (b = \text{add}(c,\ \text{tc-minus}(a))))$

THEOREM: equal-multiplep-r-new-r
$(\text{tcp}(r) \land \text{tcp}(n) \land (i \in \mathbf{N}))$
$\rightarrow \quad (\text{imultiplep}(\text{new-r}(i,\ r,\ n),\ n) = \text{imultiplep}(r,\ n))$

THEOREM: add-a-minus-a
$\text{add}(a,\ \text{tc-minus}(a)) = 0$

THEOREM: tc-minus-tc-minus-a
tc-minus (tc-minus (a)) = tc-fix (a)

THEOREM: elim-hyp-1a-3ba
(negativep (x)
$\land$ $(n \in \mathbf{N})$
$\land$ $(n \neq 0)$
$\land$ $(i \in \mathbf{N})$
$\land$ $((n * \text{twoto}\,(i - 1)) \not< \text{negative-guts}\,(x)))$
$\rightarrow$ (add (divide $(x,\ n)$, twoto $(i - 1)) \in \mathbf{N}$)

THEOREM: elim-hyp-1a-3bb
(negativep (x)
$\land$ $(n \in \mathbf{N})$
$\land$ $(n \neq 0)$
$\land$ $(i \in \mathbf{N})$
$\land$ $((n * \text{twoto}\,(i - 1)) \not< \text{negative-guts}\,(x))$
$\land$ $(\neg$ imultiplep $(x,\ n)))$
$\rightarrow$ (add (divide $(x,\ n)$, add (-1, twoto $(i - 1))) \in \mathbf{N}$)

THEOREM: elim-hyp-1a-3c
$((r \in \mathbf{N}) \land (n \in \mathbf{N}) \land (i \in \mathbf{N}) \land (n \neq 0) \land$ div-in-rangep $(r,\ n,\ i))$
$\rightarrow$ $((n * \text{twoto}\,(i - 1)) \not< \text{negative-guts}\,(\text{new-r}\,(i,\ r,\ n)))$

THEOREM: div-not-in-range-sign-new-r-1
$((r \in \mathbf{N})$
$\land$ $(n \in \mathbf{N})$
$\land$ $(i \in \mathbf{N})$
$\land$ $(n \neq 0)$
$\land$ $(\neg$ div-in-rangep $(r,\ n,\ i)))$
$\rightarrow$ (new-r $(i,\ r,\ n) \in \mathbf{N}$)

THEOREM: elim-hyp-1b-1a
$((x \in \mathbf{N}) \land (y \in \mathbf{N})) \rightarrow$ (divide $(x,\ y) \in \mathbf{N}$)

THEOREM: elim-hyp-1b-1b
$((x \in \mathbf{N}) \land (n \in \mathbf{N}) \land (x < (n * \text{twoto}\,(i - 1))))$
$\rightarrow$ negativep (add (divide $(x,\ n)$, tc-minus (twoto $(i - 1)$)))

THEOREM: elim-hyp-1b-1c
$((r \in \mathbf{N}) \land (n \in \mathbf{N}) \land (i \in \mathbf{N}) \land (n \neq 0) \land (r < (n * \text{twoto}\,(i))))$
$\rightarrow$ (new-r $(i,\ r,\ n) < (n * \text{twoto}\,(i - 1))$)

THEOREM: elim-hyp-1c
$((r \in \mathbf{N})$

$\wedge \quad (n \in \mathbf{N})$
$\wedge \quad (i \in \mathbf{N})$
$\wedge \quad (n \neq \mathtt{0})$
$\wedge \quad (i \neq \mathtt{0})$
$\wedge \quad (r \not< (n * \text{twoto}\,(i))))$
$\rightarrow \quad (\text{new-r}\,(i,\ r,\ n) \not< (n * \text{twoto}\,(i\ -\ 1)))$

THEOREM: elim-hyp-2a-4aa
$((r \in \mathbf{N})$
$\wedge \quad \text{tcp}\,(n)$
$\wedge \quad \text{negativep}\,(n)$
$\wedge \quad \text{div-in-rangep}\,(r,\ n,\ i)$
$\wedge \quad \text{imultiplep}\,(\text{new-r}\,(i,\ r,\ n),\ n))$
$\rightarrow \quad (\text{add}\,(\text{divide}\,(\text{new-r}\,(i,\ r,\ n),\ n),\ \mathtt{-1}) \in \mathbf{N})$

THEOREM: elim-hyp-2a-4ab
$((r \in \mathbf{N}) \wedge \text{tcp}\,(n) \wedge \text{negativep}\,(n) \wedge \text{div-in-rangep}\,(r,\ n,\ i))$
$\rightarrow \quad (\text{divide}\,(\text{new-r}\,(i,\ r,\ n),\ n) \in \mathbf{N})$

THEOREM: remainder-0-implies
$((a \in \mathbf{N})$
$\wedge \quad (b \in \mathbf{N})$
$\wedge \quad (c \in \mathbf{N})$
$\wedge \quad (a \neq \mathtt{0})$
$\wedge \quad ((c \ \mathbf{mod}\ a) = \mathtt{0})$
$\wedge \quad ((a * b) = c))$
$\rightarrow \quad ((c \div a) = b)$

THEOREM: elim-hyp-2a-4ba
$(\text{negativep}\,(x)$
$\wedge \quad \text{tcp}\,(n)$
$\wedge \quad \text{negativep}\,(n)$
$\wedge \quad \text{imultiplep}\,(x,\ n)$
$\wedge \quad ((\text{negative-guts}\,(n) * \text{twoto}\,(i\ -\ 1)) \not< \text{negative-guts}\,(x)))$
$\rightarrow \quad \text{negativep}\,(\text{add}\,(\text{divide}\,(x,\ n),\ \text{add}\,(\mathtt{-1},\ \text{tc-minus}\,(\text{twoto}\,(i\ -\ 1)))))$

THEOREM: equal-times-ab-c-equal-rem-ca-0
$((a \in \mathbf{N}) \wedge (b \in \mathbf{N}) \wedge (c \in \mathbf{N}) \wedge (a \neq \mathtt{0}) \wedge ((a * b) = c))$
$\rightarrow \quad ((c \ \mathbf{mod}\ a) = \mathtt{0})$

THEOREM: elim-hyp-2a-4bb
$(\text{negativep}\,(x)$
$\wedge \quad \text{tcp}\,(n)$
$\wedge \quad \text{negativep}\,(n)$
$\wedge \quad (\neg\ \text{imultiplep}\,(x,\ n))$
$\wedge \quad ((\text{negative-guts}\,(n) * \text{twoto}\,(i\ -\ 1)) \not< \text{negative-guts}\,(x)))$
$\rightarrow \quad \text{negativep}\,(\text{add}\,(\text{divide}\,(x,\ n),\ \text{tc-minus}\,(\text{twoto}\,(i\ -\ 1))))$

Theorem: elim-hyp-2a-4c

$((r \in \mathbf{N}) \wedge \text{tcp}(n) \wedge \text{negativep}(n) \wedge \text{div-in-rangep}(r, n, i))$
$\rightarrow \quad ((\text{negative-guts}(n) * \text{twoto}(i - 1)) \not< \text{negative-guts}(\text{new-r}(i, r, n)))$

Theorem: div-not-in-range-sign-new-r-2

$((r \in \mathbf{N})$
$\wedge \quad \text{tcp}(n)$
$\wedge \quad \text{negativep}(n)$
$\wedge \quad (i \in \mathbf{N})$
$\wedge \quad (\neg \text{ div-in-rangep}(r, n, i)))$
$\rightarrow \quad (\text{new-r}(i, r, n) \in \mathbf{N})$

Theorem: elim-hyp-2b-2a

$((x \in \mathbf{N}) \wedge \text{negativep}(n)) \rightarrow \text{negativep}(\text{add}(\texttt{-1}, \text{divide}(x, n)))$

Theorem: elim-hyp-2b-2b

$((x \in \mathbf{N})$
$\wedge \quad \text{negativep}(n)$
$\wedge \quad (i \in \mathbf{N})$
$\wedge \quad (x < (\text{twoto}(i - 1) * \text{negative-guts}(n))))$
$\rightarrow \quad (\text{add}(\text{divide}(x, n), \text{add}(\texttt{-1}, \text{twoto}(i - 1))) \in \mathbf{N})$

Theorem: elim-hyp-3aab-1c

$(\text{tcp}(r)$
$\wedge \quad \text{negativep}(r)$
$\wedge \quad (n \in \mathbf{N})$
$\wedge \quad (n \neq \texttt{0})$
$\wedge \quad \text{div-in-rangep}(r, n, i)$
$\wedge \quad (i \in \mathbf{N}))$
$\rightarrow \quad (\text{new-r}(i, r, n) < (\text{twoto}(i - 1) * n))$

Theorem: div-not-in-range-sign-new-r-3

$(\text{tcp}(r)$
$\wedge \quad \text{negativep}(r)$
$\wedge \quad (n \in \mathbf{N})$
$\wedge \quad (i \in \mathbf{N})$
$\wedge \quad (n \neq \texttt{0})$
$\wedge \quad (\neg \text{ div-in-rangep}(r, n, i)))$
$\rightarrow \quad \text{negativep}(\text{new-r}(i, r, n))$

Theorem: elim-hyp-3ba-3c

$(\text{tcp}(r)$
$\wedge \quad \text{negativep}(r)$
$\wedge \quad (n \in \mathbf{N})$
$\wedge \quad (n \neq \texttt{0})$

$\wedge$ $(i \in \mathbf{N})$
$\wedge$ $((\text{twoto}(i) * n) \not< \text{negative-guts}(r)))$
$\rightarrow$ $((\text{twoto}(i - 1) * n) \not< \text{negative-guts}(\text{new-r}(i, r, n)))$

THEOREM: elim-hyp-3c-3c
$(\text{tcp}(r)$
$\wedge$ $\text{negativep}(r)$
$\wedge$ $(n \in \mathbf{N})$
$\wedge$ $(n \neq 0)$
$\wedge$ $(i \in \mathbf{N})$
$\wedge$ $((n * \text{twoto}(i)) < \text{negative-guts}(r))$
$\wedge$ $(i \neq 0))$
$\rightarrow$ $((n * \text{twoto}(i - 1)) < \text{negative-guts}(\text{new-r}(i, r, n)))$

THEOREM: elim-hyp-4aa-2c
$(\text{tcp}(r)$
$\wedge$ $\text{negativep}(r)$
$\wedge$ $\text{tcp}(n)$
$\wedge$ $\text{negativep}(n)$
$\wedge$ $\text{div-in-rangep}(r, n, i)$
$\wedge$ $(i \in \mathbf{N}))$
$\rightarrow$ $(\text{new-r}(i, r, n) < (\text{twoto}(i - 1) * \text{negative-guts}(n)))$

THEOREM: div-not-in-range-sign-new-r-4
$(\text{tcp}(r)$
$\wedge$ $\text{negativep}(r)$
$\wedge$ $\text{tcp}(n)$
$\wedge$ $\text{negativep}(n)$
$\wedge$ $(i \in \mathbf{N})$
$\wedge$ $(\neg \text{div-in-rangep}(r, n, i)))$
$\rightarrow$ $\text{negativep}(\text{new-r}(i, r, n))$

THEOREM: elim-hyp-4ba-4c
$(\text{tcp}(r)$
$\wedge$ $\text{negativep}(r)$
$\wedge$ $\text{tcp}(n)$
$\wedge$ $\text{negativep}(n)$
$\wedge$ $(i \in \mathbf{N})$
$\wedge$ $((\text{twoto}(i) * \text{negative-guts}(n)) \not< \text{negative-guts}(r)))$
$\rightarrow$ $((\text{twoto}(i - 1) * \text{negative-guts}(n)) \not< \text{negative-guts}(\text{new-r}(i, r, n)))$

THEOREM: elim-hyp-4c-4c
$(\text{tcp}(r)$
$\wedge$ $\text{negativep}(r)$
$\wedge$ $\text{tcp}(n)$

$\land$ negativep (n)
$\land$ $(i \in \mathbf{N})$
$\land$ $(i \neq 0)$
$\land$ $((\text{twoto}\,(i) * \text{negative-guts}\,(n)) < \text{negative-guts}\,(r)))$
$\rightarrow$ $((\text{twoto}\,(i - 1) * \text{negative-guts}\,(n)) < \text{negative-guts}\,(\text{new-r}\,(i,\ r,\ n)))$

THEOREM: r-lessp-2n-quotient-1
$((n \neq 0) \land (r \not< n) \land (r < (2 * n))) \rightarrow ((r \div n) = 1)$

THEOREM: r-lessp-equal-2n-quotient-2-aux
$((n \in \mathbf{N})$
$\land$ $(n \neq 0)$
$\land$ $(n < r)$
$\land$ $((2 * n) \not< r)$
$\land$ $((r \textbf{ mod } n) = 0))$
$\rightarrow$ $((2 * n) = r)$

THEOREM: r-lessp-equal-2n-quotient-2
$((n \in \mathbf{N})$
$\land$ $(n \neq 0)$
$\land$ $(n < r)$
$\land$ $((2 * n) \not< r)$
$\land$ $((r \textbf{ mod } n) = 0))$
$\rightarrow$ $((r \div n) = 2)$

THEOREM: not-lessp-x-n-equal-quotient-1
$((x \in \mathbf{N})$
$\land$ $(x \neq 0)$
$\land$ $(n \in \mathbf{N})$
$\land$ $(n \neq 0)$
$\land$ $(n \not< x)$
$\land$ $((x \textbf{ mod } n) = 0))$
$\rightarrow$ $((x \div n) = 1)$

THEOREM: remainder-times-commuted
$((j * i) \textbf{ mod } i) = 0$

THEOREM: anrd-integer-ok-i-1
$(\text{tcp}\,(r) \land \text{tcp}\,(n) \land (n \neq 0) \land (i = 1))$
$\rightarrow$ $(\text{tc-to-integer}\,(\text{anrd}\,(i,\ r,\ n))$
$=$ **if** $i = 0$ **then** 0
elseif $r \in \mathbf{N}$
then if $n \in \mathbf{N}$
then if div-in-rangep $(r,\ n,\ i)$ **then** divide $(r,\ n)$
elseif $r < (n * \text{twoto}\,(i))$

then add (divide (r, n), tc-minus (twoto (i)))
else -1 **endif**
elseif div-in-rangep (r, n, i) **then** add (divide (r, n), -1)
elseif $r <$ (negative-guts (n) $*$ twoto (i))
then add (divide (r, n), add (twoto (i), -1))
else 0 **endif**
elseif $n \in \mathbf{N}$
then if div-in-rangep (r, n, i)
then if imultiplep (r, n) **then** divide (r, n)
else add (divide (r, n), -1) **endif**
elseif (n $*$ twoto (i)) $\not<$ negative-guts (r)
then if imultiplep (r, n) **then** add (divide (r, n), twoto (i))
else add (divide (r, n), add (twoto (i), -1)) **endif**
else 0 **endif**
elseif div-in-rangep (r, n, i)
then if imultiplep (r, n) **then** add (divide (r, n), -1)
else divide (r, n) **endif**
elseif (negative-guts (n) $*$ twoto (i)) $\not<$ negative-guts (r)
then if imultiplep (r, n)
then add (divide (r, n), add (tc-minus (twoto (i)), -1))
else add (divide (r, n), tc-minus (twoto (i))) **endif**
else -1 **endif**)

THEOREM: elim-hyp-2b-2c-corr
(($r \in \mathbf{N}$)
$\land$ tcp (n)
$\land$ negativep (n)
$\land$ ($r <$ (twoto (i) $*$ negative-guts (n))))
$\rightarrow$ (new-r (i, r, n) $<$ (twoto ($i - 1$) $*$ negative-guts (n)))

THEOREM: elim-hyp-2c-2c-corr
(($r \in \mathbf{N}$)
$\land$ tcp (n)
$\land$ negativep (n)
$\land$ ($i \in \mathbf{N}$)
$\land$ ($i \neq 0$)
$\land$ ($r \not<$ (twoto (i) $*$ negative-guts (n))))
$\rightarrow$ (new-r (i, r, n) $\not<$ (twoto ($i - 1$) $*$ negative-guts (n)))

THEOREM: lessp-twoto-sub1
(($i \in \mathbf{N}$) $\land$ ((n $*$ twoto (i)) $< x$)) $\rightarrow$ ((n $*$ twoto ($i - 1$)) $< x$)

THEOREM: elim-hyp-4ba-4aa-corr-aux
(($r \in \mathbf{N}$)
$\land$ ($n \in \mathbf{N}$)

$\land\quad (n \neq 0)$
$\land\quad (r \neq 0)$
$\land\quad ((r \textbf{ mod } n) = 0)$
$\land\quad ((n * i) < r)$
$\land\quad (i \neq 0)$
$\land\quad (i \in \mathbf{N}))$
$\rightarrow\quad (((r \div n) - i) \not< 1)$

THEOREM: elim-hyp-4ba-4aa-corr
(tcp (r)
$\land\quad$ negativep (r)
$\land\quad$ tcp (n)
$\land\quad$ negativep (n)
$\land\quad$ imultiplep $(r,\ n)$
$\land\quad$ ($\neg$ div-in-rangep $(r,\ n,\ i)$))
$\rightarrow\quad$ (add (divide (new-r $(i,\ r,\ n)$, n), -1) $\in \mathbf{N}$)

THEOREM: elim-hyp-4bb-4ab-corr
(tcp (r)
$\land\quad$ negativep (r)
$\land\quad$ tcp (n)
$\land\quad$ negativep (n)
$\land\quad$ ($\neg$ div-in-rangep $(r,\ n,\ i)$))
$\rightarrow\quad$ (divide (new-r $(i,\ r,\ n)$, n) $\in \mathbf{N}$)

THEOREM: anrd-integer-ok
(tcp (r) $\land$ tcp (n) $\land$ $(n \neq 0)$ $\land$ $(i \in \mathbf{N})$)
$\rightarrow\quad$ (tc-to-integer (anrd $(i,\ r,\ n)$)
$=$ **if** $i = 0$ **then** 0
elseif $r \in \mathbf{N}$
then if $n \in \mathbf{N}$
then if div-in-rangep $(r,\ n,\ i)$ **then** divide $(r,\ n)$
elseif $r <$ ($n *$ twoto (i))
then add (divide $(r,\ n)$, tc-minus (twoto (i)))
else -1 **endif**
elseif div-in-rangep $(r,\ n,\ i)$ **then** add (divide $(r,\ n)$, -1)
elseif $r <$ (negative-guts (n) $*$ twoto (i))
then add (divide $(r,\ n)$, add (twoto (i), -1))
else 0 **endif**
elseif $n \in \mathbf{N}$
then if div-in-rangep $(r,\ n,\ i)$
then if imultiplep $(r,\ n)$ **then** divide $(r,\ n)$
else add (divide $(r,\ n)$, -1) **endif**
elseif ($n *$ twoto (i)) $\not<$ negative-guts (r)
then if imultiplep $(r,\ n)$ **then** add (divide $(r,\ n)$, twoto (i))

```
                else add (divide (r, n), add (twoto (i), -1)) endif
            else 0 endif
        elseif div-in-rangep (r, n, i)
        then if imultiplep (r, n)  then add (divide (r, n), -1)
            else divide (r, n) endif
        elseif (negative-guts (n) * twoto (i)) ≮ negative-guts (r)
        then if imultiplep (r, n)
            then add (divide (r, n), add (tc-minus (twoto (i)), -1))
            else add (divide (r, n), tc-minus (twoto (i))) endif
        else -1 endif)
```

DEFINITION:
long-new-r (*r*, *noem*)
= **if** (*noem* ∈ **N**) = (*r* ∈ **N**) **then** add (*r*, tc-minus (*noem*))
else add (*r*, *noem*) **endif**

THEOREM: tcp-long-new-r
(tcp (*tel*) ∧ tcp (*noem*)) → tcp (long-new-r (*tel*, *noem*))

DEFINITION:
long-nrd (*i*, *r*, *noem*)
= **if** $i \simeq 0$ **then** BV-NIL
else bitvec ((*noem* ∈ **N**) = (long-new-r (*r*, *noem*) ∈ **N**),
long-nrd ($i - 1$, mult (2, long-new-r (*r*, *noem*)), *noem*)) **endif**

DEFINITION:
alt-long-new-r (*r*, *noem*)
= **if** (*r* ∈ **N**) = (*noem* ∈ **N**) **then** add (mult (2, *r*), tc-minus (*noem*))
else add (mult (2, *r*), *noem*) **endif**

DEFINITION:
alt-long-nrd (*i*, *r*, *noem*)
= **if** $i \simeq 0$ **then** BV-NIL
else bitvec ((*noem* ∈ **N**) = (alt-long-new-r (*r*, *noem*) ∈ **N**),
alt-long-nrd ($i - 1$, alt-long-new-r (*r*, *noem*), *noem*)) **endif**

THEOREM: alt-long-new-r-long-new-r-relate
tcp (*r*) → (alt-long-new-r (*r*, *n*) = long-new-r (mult (2, *r*), *n*))

DEFINITION:
induct-fn (*i*, *r*, *n*)
= **if** $i \simeq 0$ **then t**
else induct-fn ($i - 1$, long-new-r (mult (2, *r*), *n*), *n*) **endif**

THEOREM: alt-long-nrd-long-nrd-relate
(tcp (*r*) ∧ tcp (*n*)) → (alt-long-nrd (*i*, *r*, *n*) = long-nrd (*i*, mult (2, *r*), *n*))

DEFINITION:
bv-long-new-r $(r,\ noem,\ prev)$
= **if** bv-bit $(noem) = prev$
then csobv-bitvec (mcalu (bv-size (r),
f,
r,
$noem$,
nat-to-bv (12, 4),
nat-to-bv (5, 4),
nat-to-bv (10, 4),
f))
else csobv-bitvec (mcalu (bv-size (r),
f,
r,
$noem$,
nat-to-bv (12, 4),
nat-to-bv (10, 4),
nat-to-bv (10, 4),
t)) **endif**

DEFINITION:
bv-long-nrd $(i,\ r,\ noem,\ prev)$
= **if** $i \simeq 0$ **then** BV-NIL
else bitvec (bv-bit $(noem)$
= bv-bit (bv-long-new-r $(r,\ noem,\ prev)$),
bv-long-nrd $(i - 1$,
up-shift-1 (bv-long-new-r $(r,\ noem,\ prev)$, **f**),
$noem$,
bv-bit (bv-long-new-r $(r,\ noem,\ prev)$))) **endif**

DEFINITION:
alt-bv-long-new-r $(r,\ noem)$
= **if** bv-bit $(noem)$ = bv-bit (r)
then csobv-bitvec (mcalu (bv-size (r),
f,
up-shift-1 $(r$, **f**),
$noem$,
nat-to-bv (12, 4),
nat-to-bv (5, 4),
nat-to-bv (10, 4),
f))
else csobv-bitvec (mcalu (bv-size (r),
f,
up-shift-1 $(r$, **f**),

$noem$,
nat-to-bv (12, 4),
nat-to-bv (10, 4),
nat-to-bv (10, 4),
t)) **endif**

THEOREM: bv-size-alt-long-new-r
(bitvecp (r)
$\land$ bitvecp ($noem$)
$\land$ (bv-size (r) = bv-size ($noem$))
$\land$ evenp (bv-size (r))
$\land$ ($noem \neq$ BV-NIL))
$\rightarrow$ (bv-size (alt-bv-long-new-r (r, $noem$)) = bv-size (r))

DEFINITION:
alt-bv-long-nrd (i, r, $noem$)
= **if** $i \simeq 0$ **then** BV-NIL
else bitvec (bv-bit ($noem$) = bv-bit (alt-bv-long-new-r (r, $noem$)),
alt-bv-long-nrd ($i - 1$,
alt-bv-long-new-r (r, $noem$),
$noem$)) **endif**

THEOREM: alt-bv-long-new-r-bv-long-new-r-relate
alt-bv-long-new-r (r, n) = bv-long-new-r (up-shift-1 (r, **f**), n, bv-bit (r))

THEOREM: alt-bv-long-nrd-bv-long-nrd-relate
alt-bv-long-nrd (i, r, n) = bv-long-nrd (i, up-shift-1 (r, **f**), n, bv-bit (r))

THEOREM: equal-bit-implies-equal-numberp
((tc-to-integer (a) $\in \mathbf{N}$) = (tc-to-integer (b) $\in \mathbf{N}$))
= (bv-bit (a) = bv-bit (b))

THEOREM: equal-numberp-implies-tc-in-range-sub
((($a \in \mathbf{N}$) = ($b \in \mathbf{N}$))
$\land$ tcp (a)
$\land$ tc-in-rangep (a, n)
$\land$ tcp (b)
$\land$ tc-in-rangep (b, n))
$\rightarrow$ tc-in-rangep (add (a, tc-minus (b)), n)

THEOREM: not-equal-numberp-implies-tc-in-range-add
((($a \in \mathbf{N}$) $\neq$ ($b \in \mathbf{N}$))
$\land$ tcp (a)
$\land$ tc-in-rangep (a, n)
$\land$ tcp (b)
$\land$ tc-in-rangep (b, n))
$\rightarrow$ tc-in-rangep (add (a, b), n)

THEOREM: size-mcalu

$$
\begin{array}{l}
(\text{bitvecp}\,(a) \\
\;\wedge\;\; \text{bitvecp}\,(b) \\
\;\wedge\;\; \text{boolp}\,(byp) \\
\;\wedge\;\; \text{boolp}\,(ccin) \\
\;\wedge\;\; \text{controlep}\,(cr) \\
\;\wedge\;\; (\text{bv-size}\,(a) = \text{bv-size}\,(b)) \\
\;\wedge\;\; \text{evenp}\,(\text{bv-size}\,(a)) \\
\;\wedge\;\; (b \neq \textsc{bv-nil}) \\
\;\wedge\;\; \text{controlep}\,(cp) \\
\;\wedge\;\; \text{controlep}\,(ck)) \\
\;\rightarrow\;\; (\text{bv-size}\,(\text{csobv-bitvec}\,(\text{mcalu}\,(\text{bv-size}\,(a),\ byp,\ a,\ b,\ cp,\ ck,\ cr,\ ccin))) \\
\qquad = \quad \text{bv-size}\,(a))
\end{array}
$$

THEOREM: tc-in-range-mult-2-tc-to-int-1

$$
\begin{array}{l}
(\text{tc-to-integer}\,(tel) \in \mathbf{N}) \\
\;\rightarrow\;\; ((2 * \text{tc-to-integer}\,(tel)) < \text{twoto}\,(\text{bv-size}\,(tel)))
\end{array}
$$

THEOREM: special-francky-third-1

$$
\begin{array}{l}
((\text{abs}\,(\text{tc-to-integer}\,(noem)) \not< \text{abs}\,(\text{tc-to-integer}\,(tel))) \\
\;\wedge\;\; (\neg\ \text{tc-in-rangep}\,(\text{mult}\,(2,\ \text{tc-to-integer}\,(tel)),\ \text{bv-size}\,(tel))) \\
\;\wedge\;\; (\text{tc-to-integer}\,(noem) \in \mathbf{N}) \\
\;\wedge\;\; (\text{tc-to-integer}\,(tel) \in \mathbf{N})) \\
\;\rightarrow\;\; (\neg\ \text{tc-in-rangep}\,(\text{add}\,(\text{add}\,(\text{mult}\,(2,\ \text{tc-to-integer}\,(tel)), \\
\qquad\qquad\qquad\qquad\qquad \text{tc-minus}\,(\text{tc-to-integer}\,(noem))), \\
\qquad\qquad\qquad\qquad - \text{twoto}\,(\text{bv-size}\,(tel))), \\
\qquad\qquad\qquad \text{bv-size}\,(tel)))
\end{array}
$$

THEOREM: special-francky-third-1bis

$$
\begin{array}{l}
((\text{abs}\,(\text{tc-to-integer}\,(noem)) \not< \text{abs}\,(\text{tc-to-integer}\,(tel))) \\
\;\wedge\;\; \text{bitvecp}\,(tel) \\
\;\wedge\;\; \text{bitvecp}\,(noem) \\
\;\wedge\;\; (\neg\ \text{tc-in-rangep}\,(\text{mult}\,(2,\ \text{tc-to-integer}\,(tel)),\ \text{bv-size}\,(tel))) \\
\;\wedge\;\; (\text{tc-to-integer}\,(noem) \in \mathbf{N}) \\
\;\wedge\;\; (\text{tc-to-integer}\,(tel) \in \mathbf{N})) \\
\;\rightarrow\;\; \text{negativep}\,(\text{add}\,(\text{add}\,(\text{mult}\,(2,\ \text{tc-to-integer}\,(tel)), \\
\qquad\qquad\qquad\qquad \text{tc-minus}\,(\text{tc-to-integer}\,(noem))), \\
\qquad\qquad\qquad - \text{twoto}\,(\text{bv-size}\,(tel))))
\end{array}
$$

THEOREM: special-francky-third-2

$$
\begin{array}{l}
((\text{abs}\,(\text{tc-to-integer}\,(noem)) \not< \text{abs}\,(\text{tc-to-integer}\,(tel))) \\
\;\wedge\;\; (\neg\ \text{tc-in-rangep}\,(\text{mult}\,(2,\ \text{tc-to-integer}\,(tel)),\ \text{bv-size}\,(tel))) \\
\;\wedge\;\; \text{negativep}\,(\text{tc-to-integer}\,(noem)) \\
\;\wedge\;\; (\text{tc-to-integer}\,(tel) \in \mathbf{N})) \\
\;\rightarrow\;\; (\neg\ \text{tc-in-rangep}\,(\text{add}\,(\text{add}\,(\text{mult}\,(2,\ \text{tc-to-integer}\,(tel)),
\end{array}
$$

tc-to-integer (*noem*)),
− twoto (bv-size (*tel*))),
bv-size (*tel*)))

Theorem: special-francky-third-2-bis
((abs (tc-to-integer (*noem*)) $\not<$ abs (tc-to-integer (*tel*)))
∧ bitvecp (*tel*)
∧ bitvecp (*noem*)
∧ (¬ tc-in-rangep (mult (2, tc-to-integer (*tel*)), bv-size (*tel*)))
∧ negativep (tc-to-integer (*noem*))
∧ (tc-to-integer (*tel*) ∈ **N**))
→ negativep (add (add (mult (2, tc-to-integer (*tel*)), tc-to-integer (*noem*)),
− twoto (bv-size (*tel*))))

Theorem: special-francky-third-3
((abs (tc-to-integer (*noem*)) $\not<$ abs (tc-to-integer (*tel*)))
∧ (¬ tc-in-rangep (mult (2, tc-to-integer (*tel*)), bv-size (*tel*)))
∧ negativep (tc-to-integer (*noem*))
∧ negativep (tc-to-integer (*tel*)))
→ (¬ tc-in-rangep (add (add (mult (2, tc-to-integer (*tel*)),
tc-minus (tc-to-integer (*noem*))),
twoto (bv-size (*tel*))),
bv-size (*tel*)))

Theorem: special-francky-third-3-bis
((abs (tc-to-integer (*noem*)) $\not<$ abs (tc-to-integer (*tel*)))
∧ bitvecp (*tel*)
∧ bitvecp (*noem*)
∧ (¬ tc-in-rangep (mult (2, tc-to-integer (*tel*)), bv-size (*tel*)))
∧ negativep (tc-to-integer (*noem*))
∧ negativep (tc-to-integer (*tel*)))
→ (add (add (mult (2, tc-to-integer (*tel*)), tc-minus (tc-to-integer (*noem*))),
twoto (bv-size (*tel*))) ∈ **N**)

Theorem: special-francky-third-4
((abs (tc-to-integer (*noem*)) $\not<$ abs (tc-to-integer (*tel*)))
∧ (¬ tc-in-rangep (mult (2, tc-to-integer (*tel*)), bv-size (*tel*)))
∧ (tc-to-integer (*noem*) ∈ **N**)
∧ negativep (tc-to-integer (*tel*)))
→ (¬ tc-in-rangep (add (add (mult (2, tc-to-integer (*tel*)),
tc-to-integer (*noem*)),
twoto (bv-size (*tel*))),
bv-size (*tel*)))

Theorem: special-francky-third-4-bis

$((\text{abs}(\text{tc-to-integer}(noem)) \not< \text{abs}(\text{tc-to-integer}(tel)))$
$\land\quad \text{bitvecp}(tel)$
$\land\quad \text{bitvecp}(noem)$
$\land\quad (\neg\ \text{tc-in-rangep}(\text{mult}(2,\ \text{tc-to-integer}(tel)),\ \text{bv-size}(tel)))$
$\land\quad (\text{tc-to-integer}(noem) \in \mathbf{N})$
$\land\quad \text{negativep}(\text{tc-to-integer}(tel)))$
$\to\quad (\text{add}(\text{add}(\text{mult}(2,\ \text{tc-to-integer}(tel)),\ \text{tc-to-integer}(noem)),$
$\qquad\qquad \text{twoto}(\text{bv-size}(tel))) \in \mathbf{N})$

THEOREM: append-not-nil-2
$\text{bitvecp}(a) \to (\text{bv-append}(a,\ \text{bitvec}(x,\ \text{BV-NIL})) \neq \text{BV-NIL})$

THEOREM: negativep-mult
$(\text{tcp}(x) \land \text{negativep}(x)) \to \text{negativep}(\text{mult}(2,\ x))$

THEOREM: alt-bv-long-new-r-alt-long-new-r-relate
$(\text{bitvecp}(tel)$
$\land\quad \text{bitvecp}(noem)$
$\land\quad (\text{bv-size}(tel) = \text{bv-size}(noem))$
$\land\quad (noem \neq \text{BV-NIL})$
$\land\quad \text{evenp}(\text{bv-size}(tel))$
$\land\quad (\text{abs}(\text{tc-to-integer}(noem)) \not< \text{abs}(\text{tc-to-integer}(tel))))$
$\to\quad (\text{tc-to-integer}(\text{alt-bv-long-new-r}(tel,\ noem))$
$\qquad = \quad \text{alt-long-new-r}(\text{tc-to-integer}(tel),\ \text{tc-to-integer}(noem)))$

THEOREM: special-francky-first-leq
$(\text{tcp}(tel) \land \text{tcp}(noem) \land (\text{abs}(noem) \not< \text{abs}(tel)))$
$\to\quad (\text{abs}(noem) \not< \text{abs}(\text{alt-long-new-r}(tel,\ noem)))$

THEOREM: alt-bv-long-nrd-alt-long-nrd-relate
$(\text{bitvecp}(tel)$
$\land\quad \text{bitvecp}(n)$
$\land\quad (\text{bv-size}(tel) = \text{bv-size}(n))$
$\land\quad (n \neq \text{BV-NIL})$
$\land\quad \text{evenp}(\text{bv-size}(tel))$
$\land\quad (\text{abs}(\text{tc-to-integer}(n)) \not< \text{abs}(\text{tc-to-integer}(tel))))$
$\to\quad (\text{alt-bv-long-nrd}(i,\ tel,\ n)$
$\qquad = \quad \text{alt-long-nrd}(i,\ \text{tc-to-integer}(tel),\ \text{tc-to-integer}(n)))$

DEFINITION: $\text{new-q}(prev,\ noem) = (prev = \text{bv-bit}(noem))$

DEFINITION:
$\text{new-z}(z,\ noem,\ prev)$
$= \quad$ **if** $prev = \text{bv-bit}(noem)$
then $\text{csobv-bitvec}(\text{mcalu}(\text{bv-size}(z),$

f,
z,
noem,
nat-to-bv (`12`, `4`),
nat-to-bv (`5`, `4`),
nat-to-bv (`10`, `4`),
f))
else csobv-bitvec (mcalu (bv-size (z),
f,
z,
noem,
nat-to-bv (`12`, `4`),
nat-to-bv (`10`, `4`),
nat-to-bv (`10`, `4`),
t)) **endif**

THEOREM: size-new-z
(bitvecp (z)
∧ bitvecp (*noem*)
∧ (bv-size (*noem*) = bv-size (z))
∧ evenp (bv-size (z))
∧ (*noem* ≠ BV-NIL))
→ (bv-size (new-z (z, *noem*, *prev*)) = bv-size (z))

DEFINITION:
hard-anrd-it (i, *noem*, *ps*, z, *prev*)
= **if** bitvecp (*noem*)
∧ bitvecp (*ps*)
∧ bitvecp (z)
∧ (bv-size (*noem*) = bv-size (*ps*))
∧ (bv-size (*noem*) = bv-size (z))
∧ boolp (*prev*)
∧ ($i \in \mathbf{N}$)
then if i = `0` **then** BV-NIL
else bitvec (new-q (bv-bit (new-z (z, *noem*, *prev*)), *noem*),
hard-anrd-it ($i - 1$,
noem,
up-shift-1 (*ps*, **f**),
up-shift-1 (new-z (z, *noem*, *prev*),
bv-bit (*ps*)),
bv-bit (new-z (z, *noem*, *prev*)))) **endif**
else BV-NIL **endif**

DEFINITION:
hard-anrd-init (i, l, *tel*, *noem*)

$=$ hard-anrd-it $(i,$
$\quad$ *noem*,
$\quad$ up-shift-n $(l,\ tel,\ \mathbf{f})$,
$\quad$ do-shift-n (bv-size $(tel) - l,\ tel,$ bv-bit $(tel))$,
$\quad$ bv-bit $(tel))$

THEOREM: hack-implies-sprop
$(\mathbf{t} \rightarrow (a = b)) = (a = b)$

THEOREM: adder-with-zero-bv
bitvecp $(x) \rightarrow$ (bvco-bitvec (bv-adder $(x,$ zero-bitvec (bv-size $(x)),\ \mathbf{f})) = x)$

THEOREM: bv-append-bitvec
(bitvecp $(y) \wedge$ bitvecp $(z))$
$\rightarrow$ (bv-append (bitvec $(x,\ y),\ z)$ = bitvec $(x,$ bv-append $(y,\ z)))$

THEOREM: carry-adder-with-zero-bv
bitvecp $(x) \rightarrow (\neg$ bvco-carry (bv-adder $(x,$ zero-bitvec (bv-size $(x)),\ \mathbf{f})))$

THEOREM: bvco-carry-bit-adders-double-size
(bitvecp $(a) \wedge$ bitvecp $(b) \wedge$ bitvecp $(x) \wedge$ (bv-size (a) = bv-size $(b)))$
$\rightarrow$ (bvco-carry (bv-adder (bv-append $(a,\ x)$,
$\quad$ bv-append $(b,$ zero-bitvec (bv-size $(x)))$,
$\quad$ $\mathbf{f}))$
$\quad = $ bvco-carry (bv-adder $(a,\ b,\ \mathbf{f})))$

THEOREM: bvco-bitvec-bit-adders-double-size
(bitvecp $(a) \wedge$ bitvecp $(b) \wedge$ bitvecp $(x) \wedge$ (bv-size (a) = bv-size $(b)))$
$\rightarrow$ (bvco-bitvec (bv-adder (bv-append $(a,\ x)$,
$\quad$ bv-append $(b,$ zero-bitvec (bv-size $(x)))$,
$\quad$ $\mathbf{f}))$
$\quad = $ bv-append (bvco-bitvec (bv-adder $(a,\ b,\ \mathbf{f})),\ x))$

THEOREM: carry-adder-with-one-bv
bitvecp $(x) \rightarrow$ bvco-carry (bv-adder $(x,$ one-bitvec (bv-size $(x)),\ \mathbf{t}))$

THEOREM: adder-with-one-bv
bitvecp $(x) \rightarrow$ (bvco-bitvec (bv-adder $(x,$ one-bitvec (bv-size $(x)),\ \mathbf{t})) = x)$

THEOREM: bvco-carry-bit-adders-double-size-t
(bitvecp $(a) \wedge$ bitvecp $(b) \wedge$ bitvecp $(x) \wedge$ (bv-size (a) = bv-size $(b)))$
$\rightarrow$ (bvco-carry (bv-adder (bv-append $(a,\ x)$,
$\quad$ bv-append $(b,$ one-bitvec (bv-size $(x)))$,
$\quad$ $\mathbf{t}))$
$\quad = $ bvco-carry (bv-adder $(a,\ b,\ \mathbf{t})))$

THEOREM: bvco-bitvec-bit-adders-double-size-t
(bitvecp (a) ∧ bitvecp (b) ∧ bitvecp (x) ∧ (bv-size (a) = bv-size (b)))
→ (bvco-bitvec (bv-adder (bv-append $(a,\ x)$,
bv-append (b, one-bitvec (bv-size (x))),
t))
= bv-append (bvco-bitvec (bv-adder (a, b, **t**)), x))

THEOREM: not-zero-bv
bv-not (zero-bitvec (nb)) = one-bitvec (nb)

THEOREM: haiblnr-1
(bitvecp (ps)
∧ bitvecp (z)
∧ bitvecp (n)
∧ (bv-size (n) = bv-size (z))
∧ (bv-size (ps) = bv-size (z))
∧ ($n \neq$ BV-NIL)
∧ evenp (bv-size (z)))
→ (bv-bit (new-z (z, n, *prev*))
= bv-bit (bv-long-new-r (bv-append (z, ps),
bv-append (n, zero-bitvec (bv-size (z))),
prev)))

THEOREM: append-distributes
(bitvecp (a) ∧ bitvecp (b) ∧ bitvecp (c))
→ (bv-append (bv-append (a, b), c) = bv-append (a, bv-append (b, c)))

THEOREM: haiblnr-2
(bitvecp (z)
∧ bitvecp (n)
∧ bitvecp (ps)
∧ boolp (*prev*)
∧ (bv-size (z) = bv-size (n))
∧ (bv-size (z) = bv-size (ps))
∧ ($n \neq$ BV-NIL)
∧ evenp (bv-size (z)))
→ (bv-append (up-shift-1 (new-z (z, n, *prev*), bv-bit (ps)), up-shift-1 (ps, **f**))
= up-shift-1 (bv-long-new-r (bv-append (z, ps),
bv-append (n, zero-bitvec (bv-size (z))),
prev),
f))

THEOREM: hard-anrd-it-bv-long-nrd-relate
(bitvecp (ps)
∧ bitvecp (z)

$\wedge$ bitvecp (n)
$\wedge$ (bv-size (n) = bv-size (z))
$\wedge$ (bv-size (ps) = bv-size (z))
$\wedge$ $(n \neq$ BV-NIL$)$
$\wedge$ evenp (bv-size (z))
$\wedge$ boolp $(prev)$
$\wedge$ $(i \in \mathbf{N})$)
$\rightarrow$ (hard-anrd-it $(i,\ n,\ ps,\ z,\ prev)$
  = bv-long-nrd $(i,$
    bv-append $(z,\ ps)$,
    bv-append $(n,$ zero-bitvec (bv-size (z))),
    $prev$))

THEOREM: numberp-mult
(tcp $(a) \wedge (b \in \mathbf{N}) \wedge (b \neq 0)) \rightarrow (($mult $(a,\ b) \in \mathbf{N}) = (a \in \mathbf{N}))$

THEOREM: long-new-r-new-r-relate
$((i \in \mathbf{N}) \wedge$ tcp $(r) \wedge$ tcp $(noem))$
$\rightarrow$ (long-new-r $(r,$ mult $(noem,$ twoto $(i - 1)))$ = new-r $(i,\ r,\ noem))$

THEOREM: not-lessp-not-lessp-times
$((x \in \mathbf{N}) \wedge (x \neq 0) \wedge (y \not< z)) \rightarrow ((x * y) \not< (x * z))$

THEOREM: lessp-not-lessp-times
$(y < w) \rightarrow ((v * w) \not< (v * y))$

THEOREM: mult-add-distributes
(tcp $(x) \wedge$ tcp $(y) \wedge$ tcp (z))
$\rightarrow$ (mult $(x,$ add $(y,\ z))$ = add (mult $(x,\ y)$, mult $(x,\ z)$))

THEOREM: long-new-r-mult-2
(tcp $(r) \wedge$ tcp (n))
$\rightarrow$ (long-new-r (mult $(2,\ r)$, mult $(2,\ n)$) = mult $(2,$ long-new-r $(r,\ n)$))

THEOREM: long-nrd-mult-2
(tcp $(r) \wedge$ tcp (n))
$\rightarrow$ (long-nrd $(i,$ mult $(2,\ r)$, mult $(2,\ n)$) = long-nrd $(i,\ r,\ n)$)

THEOREM: long-nrd-anrd-relate
(tcp $(tel) \wedge$ tcp $(noem)$)
$\rightarrow$ (anrd $(i,\ tel,\ noem)$ = long-nrd $(i,$ mult $(2,\ tel)$, mult $(noem,$ twoto (i))))

DEFINITION:
simple-induct (x)
= **if** $x \simeq 0$ **then t**
  else simple-induct $(x - 1)$ **endif**

THEOREM: up-shift-1-append
(bitvecp (x) ∧ bitvecp (y) ∧ $(y \neq$ BV-NIL$)$ ∧ boolp $(sin))$
→ (up-shift-1 (bv-append $(x,\ y)$, sin)
= bv-append (up-shift-1 (x, bv-bit (y)), up-shift-1 $(y,\ sin)$))

THEOREM: append-butlast-last
(bitvecp (y) ∧ $(y \neq$ BV-NIL$)$)
→ (bv-append (butlast (y), bitvec (last (y), BV-NIL)) $= y$)

THEOREM: us1-dos-1
(bitvecp (y) ∧ boolp $(dsin)$)
→ (up-shift-1 (do-shift-1 $(y,\ dsin)$, last (y)) $= y$)

DEFINITION:
id-bitvec $(size,\ bit)$
= **if** $size \in \mathbf{N}$
then if $size \simeq 0$ **then** BV-NIL
else bitvec (bit, id-bitvec $(size - 1,\ bit)$) **endif**
else BV-NIL **endif**

THEOREM: size-id-bitvec
bv-size (id-bitvec $(size,\ bit)$) = fix $(size)$

DEFINITION:
first $(i,\ x)$
= **if** $i \simeq 0$ **then** BV-NIL
else bitvec (bv-bit (x), first ($i - 1$, bv-vec (x))) **endif**

THEOREM: first-size-x
bitvecp (x) → (first (bv-size (x), x) $= x$)

THEOREM: append-not-nil-3
bitvecp (a) → (bv-append (a, bitvec $(x,\ b)$) $\neq$ BV-NIL)

THEOREM: butlast-bitvec
(bitvecp $(rest)$ ∧ $(rest \neq$ BV-NIL$)$)
→ (butlast (bitvec $(bit,\ rest)$) = bitvec (bit, butlast $(rest)$))

THEOREM: butlast-first
(bitvecp (x) ∧ $(n \in \mathbf{N})$) → (butlast (first $(n,\ x)$) = first $(n - 1,\ x)$)

THEOREM: do-shift-n-first-append-relate
(bitvecp (x) ∧ (bv-size $(x) \not< i$) ∧ $(i \in \mathbf{N})$ ∧ boolp (bit))
→ (do-shift-n $(i,\ x,\ bit)$
= bv-append (id-bitvec $(i,\ bit)$, first (bv-size $(x) - i$, x)))

THEOREM: full-down-shift
$\text{boolp}(bit) \rightarrow (\text{do-shift-n}(\text{bv-size}(x),\ x,\ bit) = \text{id-bitvec}(\text{bv-size}(x),\ bit))$

THEOREM: up-shift-strict
$(x = \text{BV-NIL}) \rightarrow (\text{up-shift-n}(m,\ x,\ sin) = \text{BV-NIL})$

THEOREM: up-shift-1-bv-vec
$\text{bv-vec}(\text{up-shift-n}(n,\ x,\ sin)) = \text{up-shift-n}(n,\ \text{bv-vec}(x),\ sin)$

THEOREM: first-strict
$(\text{first}(n,\ x) = \text{BV-NIL}) = (n \simeq 0)$

THEOREM: up-shift-strict-2
$(\text{up-shift-n}(n,\ x,\ sin) = \text{BV-NIL})$
$= \quad ((\neg\ \text{boolp}(sin)) \vee (\neg\ \text{bitvecp}(x)) \vee (n \notin \mathbf{N}) \vee (x = \text{BV-NIL}))$

THEOREM: last-first
$(\text{boolp}(sin) \wedge (m \in \mathbf{N}) \wedge (m \neq 0) \wedge \text{bitvecp}(x) \wedge (\text{bv-size}(x) \not< m))$
$\rightarrow \quad (\text{last}(\text{first}(m,\ x)) = \text{bv-bit}(\text{up-shift-n}(m - 1,\ x,\ sin)))$

THEOREM: last-id-bitvec
$(\text{boolp}(bit) \wedge (n \not\simeq 0)) \rightarrow (\text{last}(\text{id-bitvec}(n,\ bit)) = bit)$

THEOREM: bit-us-last-dos
$(\text{bitvecp}(x) \wedge (m \in \mathbf{N}) \wedge (\text{bv-size}(x) \not< m))$
$\rightarrow \quad (\text{last}(\text{do-shift-n}(\text{bv-size}(x) - m,\ x,\ \text{bv-bit}(x)))$
$\quad\quad = \quad \text{bv-bit}(\text{up-shift-n}(m - 1,\ x,\ \mathbf{f})))$

THEOREM: append-of-up-shift
$(\text{bitvecp}(x) \wedge (l \in \mathbf{N}) \wedge (\text{bv-size}(x) = nb) \wedge (nb \not< l))$
$\rightarrow \quad (\text{bv-append}(\text{do-shift-n}(nb - l,\ x,\ \text{bv-bit}(x)),\ \text{up-shift-n}(l,\ x,\ \mathbf{f}))$
$\quad\quad = \quad \text{up-shift-n}(l,\ \text{bv-append}(\text{id-bitvec}(nb,\ \text{bv-bit}(x)),\ x),\ \mathbf{f}))$

THEOREM: id-bitvec-strict
$(\text{id-bitvec}(n,\ bit) = \text{BV-NIL}) = (n \simeq 0)$

THEOREM: tc-to-int-simple-sign-extension
$\text{tc-to-integer}(\text{bitvec}(\text{bv-bit}(x),\ x)) = \text{tc-to-integer}(x)$

THEOREM: tc-to-int-sign-extension
$\text{tc-to-integer}(\text{bv-append}(\text{id-bitvec}(n,\ \text{bv-bit}(x)),\ x)) = \text{tc-to-integer}(x)$

THEOREM: mult-1
$\text{tcp}(x) \rightarrow (\text{mult}(1,\ x) = x)$

THEOREM: zero-bitvec-extension
$\text{bitvec}(\mathbf{f},\ \text{zero-bitvec}(n)) = \text{bv-append}(\text{zero-bitvec}(n),\ \text{bitvec}(\mathbf{f},\ \text{BV-NIL}))$

THEOREM: tc-to-int-append-simple-zero-bitvec
$\text{tc-to-integer}(\text{bv-append}(x,\ \text{bitvec}(\mathbf{f},\ \text{BV-NIL}))) = \text{mult}(\mathtt{2},\ \text{tc-to-integer}(x))$

THEOREM: tc-to-int-append-zero-bv
$(n \in \mathbf{N})$
$\rightarrow\quad (\text{tc-to-integer}(\text{bv-append}(x,\ \text{zero-bitvec}(n)))$
$\quad\quad = \quad \text{mult}(\text{twoto}(n),\ \text{tc-to-integer}(x)))$

THEOREM: times-conserves-lessp
$((a \in \mathbf{N}) \wedge (b \in \mathbf{N}) \wedge (c \in \mathbf{N}) \wedge (b \not\simeq \mathtt{0}) \wedge (c \not< a))$
$\rightarrow\quad ((b * c) \not< a)$

THEOREM: times-conserves-lessp-dual
$((a \in \mathbf{N}) \wedge (b \in \mathbf{N}) \wedge (c \in \mathbf{N}) \wedge (c \not\simeq \mathtt{0}) \wedge (b \not< a))$
$\rightarrow\quad ((b * c) \not< a)$

THEOREM: lessp-abs-max-upshift
$(\text{tcp}(a) \wedge \text{tcp}(b) \wedge \text{tcp}(c) \wedge (\text{abs}(b) \not< \text{abs}(a)) \wedge (c \neq \mathtt{0}))$
$\rightarrow\quad (\text{abs}(\text{mult}(b,\ c)) \not< \text{abs}(a))$

THEOREM: abs-upper-bound-tc-to-int
$\text{twoto}(\text{bv-size}(x)) \not< \text{abs}(\text{tc-to-integer}(x))$

THEOREM: hard-anrd-anrd-relate-1
$(\text{bitvecp}(tel)$
$\wedge\quad \text{bitvecp}(noem)$
$\wedge\quad (\text{bv-size}(tel) = \text{bv-size}(noem))$
$\wedge\quad (\text{bv-size}(tel) = nb)$
$\wedge\quad \text{evenp}(\text{bv-size}(tel))$
$\wedge\quad (noem \neq \text{BV-NIL})$
$\wedge\quad (\text{tc-to-integer}(noem) \neq \mathtt{0})$
$\wedge\quad (nb \not< i)$
$\wedge\quad (i \in \mathbf{N}))$
$\rightarrow\quad (\text{hard-anrd-init}(i,\ \mathtt{1},\ tel,\ noem)$
$\quad\quad = \quad \text{anrd}(i,$
$\quad\quad\quad\quad \text{tc-to-integer}(tel),$
$\quad\quad\quad\quad \text{mult}(\text{tc-to-integer}(noem),\ \text{twoto}(nb - i))))$

THEOREM: times-conserves-lessp-real
$((a \in \mathbf{N}) \wedge (b \in \mathbf{N}) \wedge (c \in \mathbf{N}) \wedge (b \not\simeq \mathtt{0}) \wedge (a < c))$
$\rightarrow\quad (a < (b * c))$

THEOREM: times-conserves-lessp-dual-real
$((a \in \mathbf{N}) \wedge (b \in \mathbf{N}) \wedge (c \in \mathbf{N}) \wedge (b \not\simeq \mathtt{0}) \wedge (a < c))$
$\rightarrow\quad (a < (c * b))$

THEOREM: neg-guts-tc-to-integer-0
(negative-guts (tc-to-integer (a)) = 0) = (tc-to-integer (a) $\in$ **N**)

THEOREM: div-in-rangep-tc-to-ints
(bitvecp (*tel*)
 $\wedge$ bitvecp (*noem*)
 $\wedge$ (bv-size (*tel*) = bv-size (*noem*))
 $\wedge$ (tc-to-integer (*noem*) $\neq$ 0))
$\rightarrow$ div-in-rangep (tc-to-integer (*tel*), tc-to-integer (*noem*), bv-size (*tel*))

THEOREM: hard-anrd-init-divides
(bitvecp (*tel*)
 $\wedge$ bitvecp (*noem*)
 $\wedge$ (bv-size (*tel*) = bv-size (*noem*))
 $\wedge$ (bv-size (*tel*) = *nb*)
 $\wedge$ evenp (bv-size (*tel*))
 $\wedge$ (*noem* $\neq$ BV-NIL)
 $\wedge$ (tc-to-integer (*noem*) $\neq$ 0))
$\rightarrow$ (tc-to-integer (hard-anrd-init (*nb*, 1, *tel*, *noem*))
 = **if** tc-to-integer (*tel*) $\in$ **N**
 then if tc-to-integer (*noem*) $\in$ **N**
 then divide (tc-to-integer (*tel*), tc-to-integer (*noem*))
 else add (divide (tc-to-integer (*tel*),
 tc-to-integer (*noem*)),
 -1) **endif**
 elseif tc-to-integer (*noem*) $\in$ **N**
 then if imultiplep (tc-to-integer (*tel*), tc-to-integer (*noem*))
 then divide (tc-to-integer (*tel*), tc-to-integer (*noem*))
 else add (divide (tc-to-integer (*tel*),
 tc-to-integer (*noem*)),
 -1) **endif**
 elseif imultiplep (tc-to-integer (*tel*), tc-to-integer (*noem*))
 then add (divide (tc-to-integer (*tel*), tc-to-integer (*noem*)), -1)
 else divide (tc-to-integer (*tel*), tc-to-integer (*noem*)) **endif**)

;;))

Index