

```

;
; 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  
 $(y \notin \mathbf{N}) \rightarrow ((x + y) = \text{fix}(x))$

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

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

$$((a + b) = 0) = ((a \simeq 0) \wedge (b \simeq 0))$$

THEOREM: plus-cancellation

$$((a + b) = (a + c)) = (\text{fix}(b) = \text{fix}(c))$$

THEOREM: times-zero2

$$(y \notin \mathbf{N}) \rightarrow ((x * y) = 0)$$

THEOREM: distributivity-of-times-over-plus

$$(x * (y + z)) = ((x * y) + (x * z))$$

THEOREM: times-add1

$$\begin{aligned} &(x * (1 + y)) \\ &= \text{if } y \in \mathbf{N} \text{ then } x + (x * y) \\ &\quad \text{else fix}(x) \text{ endif} \end{aligned}$$

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

$$((x * y) = 0) = ((x \simeq 0) \vee (y \simeq 0))$$

THEOREM: times-1

$$(1 * x) = \text{fix}(x)$$

THEOREM: equal-bools

$$\begin{aligned} &(((bool1 = \mathbf{t}) \vee (bool1 = \mathbf{f})) \wedge ((bool2 = \mathbf{t}) \vee (bool2 = \mathbf{f}))) \\ &\rightarrow ((bool1 = bool2) = ((bool1 \rightarrow bool2) \wedge (bool2 \rightarrow bool1))) \end{aligned}$$

EVENT: Disable equal-bools.

THEOREM: lessp-times

$$((y * x) < (x * z)) = ((x \neq 0) \wedge (y < z))$$

THEOREM: times-2-not-1

$$(2 * x) \neq 1$$

DEFINITION:

twoto( $n$ )

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

$$((y \in \mathbf{N}) \wedge (y \not\prec x)) \rightarrow ((x + (y - x)) = y)$$

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

$$\begin{aligned} & ((x + (y - z)) - y) \\ &= \text{if } y < z \text{ then } x - y \\ &  \text{else } x - z \text{ endif} \end{aligned}$$

THEOREM: difference-difference

$$((x - y) - z) = (x - (y + z))$$

THEOREM: lessp-difference

$$((x - y) < x) = ((x \neq 0) \wedge (y \neq 0))$$

THEOREM: difference-add1

$$\begin{aligned} & ((1 + x) - y) \\ = & \text{if } y < (1 + x) \text{ then } 1 + (x - y) \\ & \text{else } 0 \text{ endif} \end{aligned}$$

THEOREM: remainder-quotient

$$((x \bmod y) + (y * (x \div y))) = \text{fix}(x)$$

THEOREM: remainder-by-nonnumber

$$(x \notin \mathbf{N}) \rightarrow ((y \bmod x) = \text{fix}(y))$$

THEOREM: lessp-remainder

$$((x \bmod y) < y) = (y \neq 0)$$

THEOREM: remainder-quotient-elim

$$((y \neq 0) \wedge (x \in \mathbf{N})) \rightarrow (((x \bmod y) + (y * (x \div y))) = x)$$

THEOREM: remainder-x-x

$$(x \bmod x) = 0$$

THEOREM: remainder-plus

$$((j + x) \bmod j) = (x \bmod j)$$

THEOREM: remainder-plus-times

$$((x + (i * j)) \bmod j) = (x \bmod j)$$

THEOREM: remainder-plus-times-commuted

$$((x + (j * i)) \bmod j) = (x \bmod j)$$

THEOREM: remainder-times

$$((j * i) \bmod j) = 0$$

THEOREM: quotient-plus-times

$$\begin{aligned} & ((x + (i * j)) \div j) \\ = & \text{if } j \simeq 0 \text{ then } 0 \\ & \text{else } i + (x \div j) \text{ endif} \end{aligned}$$

THEOREM: quotient-plus-times-commuted

$$\begin{aligned} & ((x + (j * i)) \div j) \\ = & \text{if } j \simeq 0 \text{ then } 0 \\ & \text{else } i + (x \div j) \text{ endif} \end{aligned}$$

THEOREM: quotient-times

$$\begin{aligned} & ((j * i) \div j) \\ &= \text{if } j \simeq 0 \text{ then } 0 \\ & \quad \text{else fix } (i) \text{ endif} \end{aligned}$$

EVENT: Disable times.

THEOREM: times-distributes-over-remainder

$$((x * y) \bmod (x * z)) = (x * (y \bmod z))$$

THEOREM: remainder-of-1

$$\begin{aligned} & (1 \bmod x) \\ &= \text{if } x = 1 \text{ then } 0 \\ & \quad \text{else } 1 \text{ endif} \end{aligned}$$

THEOREM: remainder-crock3

$$\begin{aligned} & ((x + (y - z)) \bmod y) \\ &= \text{if } (x + (y - z)) < y \text{ then } x + (y - z) \\ & \quad \text{elseif } y < z \text{ then } x \bmod y \\ & \quad \text{else } (x - z) \bmod y \text{ endif} \end{aligned}$$

THEOREM: remainder-of-0

$$(0 \bmod x) = 0$$

THEOREM: remainder-crock4

$$\begin{aligned} & ((1 + (x + (y - z))) \bmod y) \\ &= \text{if } (1 + (x + (y - z))) < y \text{ then } 1 + (x + (y - z)) \\ & \quad \text{elseif } y < z \text{ then } (1 + x) \bmod y \\ & \quad \text{elseif } z < (1 + x) \text{ then } (1 + (x - z)) \bmod y \\ & \quad \text{else } 0 \text{ endif} \end{aligned}$$

DEFINITION:

evenp ( $n$ )

$$\begin{aligned} &= \text{if } n \in \mathbf{N} \\ & \quad \text{then if } n \simeq 0 \text{ then t} \\ & \quad \quad \text{elseif } n = 1 \text{ then f} \\ & \quad \quad \text{else evenp}((n - 1) - 1) \text{ endif} \\ & \quad \text{else f endif} \end{aligned}$$

THEOREM: evenp-add1

$$((a \in \mathbf{N}) \wedge \text{evenp}(a)) \rightarrow (\neg \text{evenp}(1 + a))$$

THEOREM: not-evenp-add1

$$((a \in \mathbf{N}) \wedge (\neg \text{evenp}(a))) \rightarrow \text{evenp}(1 + a)$$

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:  
 $\text{carry}(c)$   
 $=$  **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:  
 $\text{bv-size}(a)$   
 $=$  **if**  $\text{bitvecp}(a)$   
    **then if**  $a = \text{BV-NIL}$  **then** 0  
        **else**  $1 + \text{bv-size}(\text{bv-vec}(a))$  **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:

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

DEFINITION:

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

DEFINITION:

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

THEOREM:  $\text{bv-exor-nil-a}$

$(a = \text{BV-NIL}) \rightarrow (\text{bv-exor}(a, b) = \text{BV-NIL})$

THEOREM:  $\text{bv-exor-nil-b}$

$(b = \text{BV-NIL}) \rightarrow (\text{bv-exor}(a, b) = \text{BV-NIL})$

DEFINITION:

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

DEFINITION:

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

THEOREM:  $\text{size-zerobv}$

$\text{bv-size}(\text{zero-bitvec}(\text{bv-size}(a))) = \text{bv-size}(a)$

THEOREM: size-onebv  
 $\text{bv-size}(\text{one-bitvec}(\text{bv-size}(a))) = \text{bv-size}(a)$

THEOREM: size-bvnot  
 $\text{bv-size}(\text{bv-not}(a)) = \text{bv-size}(a)$

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

THEOREM: size-bvinv  
 $\text{bv-size}(\text{bv-invert-even}(a)) = \text{bv-size}(a)$

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

THEOREM: boolp-bvco-ovf  
 $\text{boolp}(\text{bvco-ovf}(a))$

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:  
 $\text{tcp}(x) = ((x \in \mathbf{N}) \vee (\text{negativep}(x) \wedge (\text{negative-guts}(x) \neq 0)))$



DEFINITION:

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

DEFINITION:

add( $a, b$ )  
= **if** negativep( $a$ )  
    **then if** negativep( $b$ ) **then**  $-(\text{negative-guts}(a) + \text{negative-guts}(b))$   
        **elseif**  $b < \text{negative-guts}(a)$  **then**  $-(\text{negative-guts}(a) - b)$   
        **else**  $b - \text{negative-guts}(a)$  **endif**  
    **elseif** negativep( $b$ )  
    **then if**  $a < \text{negative-guts}(b)$  **then**  $-(\text{negative-guts}(b) - a)$   
        **else**  $a - \text{negative-guts}(b)$  **endif**  
    **else**  $a + b$  **endif**

THEOREM: commutativity2-of-add

add( $x, \text{add}(y, z)$ ) = add( $y, \text{add}(x, z)$ )

THEOREM: commutativity-of-add

add( $x, y$ ) = add( $y, x$ )

THEOREM: associativity-of-add

add(add( $x, y$ ),  $z$ ) = add( $x, \text{add}(y, z)$ )

DEFINITION:

bv-to-nat( $a$ )  
= **if** bitvecp( $a$ )  
    **then if**  $a = \text{BV-NIL}$  **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( $a, \text{size}$ )  
= **if**  $\text{size} \simeq 0$  **then** BV-NIL  
    **else** bitvec(**if** ( $a \div \text{twoto}(\text{size} - 1)$ ) = 1 **then** t  
        **else** f **endif**,  
        nat-to-bv( $a \bmod \text{twoto}(\text{size} - 1), \text{size} - 1)$ ) **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-range-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 ( $x$ ,  $n$ )))

$$\begin{aligned}
& + \text{integer-to-nat}(y, n) \\
& \mathbf{mod} \text{ twoto}(n), \\
& n) \\
= & \mathbf{if} \text{ tc-in-range}(\text{add}(x, \text{add}(y, \text{carry}(c))), n) \\
& \mathbf{then} \text{ add}(x, \text{add}(y, \text{carry}(c))) \\
& \mathbf{elseif} \text{ negative}(\text{add}(x, \text{add}(y, \text{carry}(c)))) \\
& \mathbf{then} \text{ add}(x, \text{add}(y, \text{add}(\text{carry}(c), \text{twoto}(n)))) \\
& \mathbf{else} \text{ add}(x, \text{add}(y, \text{add}(\text{carry}(c), -\text{twoto}(n)))) \mathbf{endif}
\end{aligned}$$

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  
 $\text{tc-to-integer}(a) = \text{nat-to-integer}(\text{bv-to-nat}(a), \text{bv-size}(a))$

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

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

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

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

DEFINITION:

$\text{bv-adder}(a, b, \text{cin})$   
 $= \mathbf{if} \text{ bitvec}p(a)$   
 $\quad \wedge \text{ bitvec}p(b)$   
 $\quad \wedge (\text{bv-size}(a) = \text{bv-size}(b))$   
 $\quad \wedge \text{ bool}p(\text{cin})$   
 $\mathbf{then if } a = \text{BV-NIL} \mathbf{ then } \text{bitvec-carry-ovf}(\text{BV-NIL}, \text{cin}, \mathbf{f})$   
 $\quad \mathbf{else } \text{bitvec-carry-ovf}(\text{bitvec}(\text{exor}(\text{exor}(\text{bv-bit}(a), \text{bv-bit}(b)),$   
 $\quad \quad \text{bvco-carry}(\text{bv-adder}(\text{bv-vec}(a),$   
 $\quad \quad \quad \text{bv-vec}(b),$   
 $\quad \quad \quad \text{cin}))),$   
 $\quad \text{bvco-bitvec}(\text{bv-adder}(\text{bv-vec}(a),$   
 $\quad \quad \text{bv-vec}(b),$   
 $\quad \quad \text{cin}))),$   
 $(\text{bv-bit}(a) \wedge \text{bv-bit}(b))$

$$\begin{aligned}
& \vee \text{ (bv-bit } (a) \\
& \quad \wedge \text{ bvco-carry (bv-adder (bv-vec } (a), \\
& \quad \quad \quad \text{bv-vec } (b), \\
& \quad \quad \quad \text{cin}))} \\
& \vee \text{ (bv-bit } (b) \\
& \quad \wedge \text{ bvco-carry (bv-adder (bv-vec } (a), \\
& \quad \quad \quad \text{bv-vec } (b), \\
& \quad \quad \quad \text{cin}))}, \\
& \text{bvco-carry (bv-adder (bv-vec } (a), \\
& \quad \quad \quad \text{bv-vec } (b), \\
& \quad \quad \quad \text{cin}))} \text{ endif}
\end{aligned}$$

**else BVCO-NIL endif**

THEOREM: size-bv-adder

$$\begin{aligned}
& (\text{bitvecp } (a) \wedge \text{bitvecp } (b) \wedge \text{boolp } (cin) \wedge (\text{bv-size } (a) = \text{bv-size } (b))) \\
& \rightarrow (\text{bv-size (bvco-bitvec (bv-adder } (a, b, cin))) = \text{bv-size } (a))
\end{aligned}$$

THEOREM: bv-adder-plusses

$$\begin{aligned}
& (\text{bitvecp } (a) \wedge \text{bitvecp } (b) \wedge \text{boolp } (cin) \wedge (\text{bv-size } (a) = \text{bv-size } (b))) \\
& \rightarrow (\text{bv-to-nat (bitvec (bvco-carry (bv-adder } (a, b, cin)), \\
& \quad \quad \quad \text{bvco-bitvec (bv-adder } (a, b, cin)))) \\
& \quad = (\text{bv-to-nat } (a) + \text{bv-to-nat } (b) + \text{carry } (cin))
\end{aligned}$$

THEOREM: bv-adder-non-nil

$$\begin{aligned}
& (\text{bitvecp } (a) \\
& \quad \wedge \text{bitvecp } (b) \\
& \quad \wedge (\text{bv-size } (a) = \text{bv-size } (b)) \\
& \quad \wedge \text{boolp } (cin) \\
& \quad \wedge (a \neq \text{BV-NIL}) \\
& \quad \wedge (b \neq \text{BV-NIL})) \\
& \rightarrow ((\text{bvco-bitvec (bv-adder } (a, b, cin)) = \text{BV-NIL}) = \mathbf{f})
\end{aligned}$$

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

$$\begin{aligned}
& (\text{bitvecp } (a) \wedge \text{bitvecp } (b) \wedge (\text{bv-size } (a) = \text{bv-size } (b)) \wedge \text{boolp } (cin)) \\
& \rightarrow (\text{bv-to-nat (bvco-bitvec (bv-adder } (a, b, cin))) \\
& \quad = ((\text{bv-to-nat } (a) + \text{bv-to-nat } (b) + \text{carry } (cin)) \\
& \quad \quad \mathbf{mod} \text{ twoto (bv-size } (a))))
\end{aligned}$$

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

THEOREM: integer-interpretation-of-bv-adder-output-lemma1

$$\begin{aligned}
& (\text{bitvecp } (a) \wedge \text{bitvecp } (b) \wedge (\text{bv-size } (a) = \text{bv-size } (b)) \wedge \text{boolp } (cin)) \\
& \rightarrow (\text{tc-to-integer (bvco-bitvec (bv-adder } (a, b, cin))) \\
& \quad = \text{nat-to-integer } ((\text{bv-to-nat } (a) + \text{bv-to-nat } (b) + \text{carry } (cin)) \\
& \quad \quad \mathbf{mod} \text{ twoto (bv-size } (a)), \\
& \quad \quad \text{bv-size } (a)))
\end{aligned}$$

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

```

(bitvecp (a)
  ∧ bitvecp (b)
  ∧ (a ≠ BV-NIL)
  ∧ boolp (cin)
  ∧ (b ≠ BV-NIL)
  ∧ (bv-size (a) = bv-size (b)))
→ (tc-to-integer (bvco-bitvec (bv-adder (a, b, cin)))
   = if tc-in-rangep (add (tc-to-integer (a),
                          add (tc-to-integer (b), carry (cin))),
                      bv-size (a))
   then add (tc-to-integer (a), add (tc-to-integer (b), carry (cin)))
   elseif negativep (add (tc-to-integer (a),
                         add (tc-to-integer (b), carry (cin))))
   then add (tc-to-integer (a),
            add (tc-to-integer (b),
                add (carry (cin), twoto (bv-size (a)))))
   else add (tc-to-integer (a),
            add (tc-to-integer (b),
                add (carry (cin), - twoto (bv-size (a))))) endif)

```

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

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

DEFINITION:  
alucod-fc (*ik*, *ip*, *cc*) = ((*ik* ∧ (¬ *ip*)) ∨ (*ip* ∧ (¬ *cc*)))

THEOREM: boolp-alucod-fc  
boolp (alucod-fc (*k*, *p*, *c*))

DEFINITION:  
alucev-fc (*ik*, *ip*, *cc*) = (((¬ *ik*) ∧ (¬ *ip*)) ∨ (*ip* ∧ (¬ *cc*)))

THEOREM: boolp-alucev-fc  
boolp (alucev-fc (*k*, *p*, *c*))

DEFINITION:  
addbyp-fcout (*cinshot*, *prop1*, *prop2*, *prop3*, *prop4*, *cinnorm*)  
= **if** *prop1* ∧ *prop2* ∧ *prop3* ∧ *prop4* **then** *cinshot*  
**else** *cinnorm* **endif**

DEFINITION:  
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)

```

$$\begin{aligned}
& \wedge \text{ boolp}(cp3) \\
& \wedge \text{ boolp}(cp4) \\
& \wedge \text{ boolp}(cin) \\
\rightarrow & \text{ (addbyp-fcout}(cin, \\
& \quad cp4, \\
& \quad cp3, \\
& \quad cp2, \\
& \quad cp1, \\
& \quad \text{alucev-fc}(ck1, \\
& \quad \quad cp1, \\
& \quad \quad \text{alucod-fc}(ck2, \\
& \quad \quad \quad cp2, \\
& \quad \quad \quad \text{alucev-fc}(ck3, \\
& \quad \quad \quad \quad cp3, \\
& \quad \quad \quad \quad \text{alucod-fc}(ck4, cp4, cin)))))) \\
= & \text{ alucev-fc}(ck1, \\
& \quad cp1, \\
& \quad \text{alucod-fc}(ck2, \\
& \quad \quad cp2, \\
& \quad \quad \text{alucev-fc}(ck3, cp3, \text{alucod-fc}(ck4, cp4, cin))))))
\end{aligned}$$

THEOREM: c4-c4byp-relate

$$\begin{aligned}
& (\text{boolp}(p1) \\
& \wedge \text{ boolp}(p2) \\
& \wedge \text{ boolp}(p3) \\
& \wedge \text{ boolp}(p4) \\
& \wedge \text{ boolp}(k1) \\
& \wedge \text{ boolp}(k2) \\
& \wedge \text{ boolp}(k3) \\
& \wedge \text{ boolp}(k4) \\
& \wedge \text{ boolp}(cin)) \\
\rightarrow & (\text{c4byp}(p1, p2, p3, p4, k1, k2, k3, k4, cin) \\
& = \text{c4}(p1, p2, p3, p4, k1, k2, k3, k4, cin))
\end{aligned}$$

EVENT: Disable c4.

EVENT: Disable c4byp.

EVENT: Disable c4-c4byp-help.

DEFINITION:

$$\begin{aligned}
& \text{carry-col}(ik, ip, ccin) \\
= & \text{ if bitvecp}(ik)
\end{aligned}$$



```

    ∧ 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  
 $(\text{bitvecp}(a) \wedge \text{bitvecp}(b) \wedge (\text{bv-size}(a) = \text{bv-size}(b)) \wedge \text{boolp}(ccin))$   
 $\rightarrow (\text{bv-size}(\text{bvco-bitvec}(\text{carry-col}(a, b, ccin))) = \text{bv-size}(a))$

DEFINITION:  $\text{up}(nb) = ((4 * ((nb - 10) \div 4)) + 9)$

DEFINITION:  
 $\text{carry-col2}(ik, ip, ccin, nb)$   
 $=$  **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))))),
                                bv-vec (bv-vec (bv-vec (ik)))),
                                bv-vec (bv-vec (bv-vec (ip)))),
                                ccin,
                                nb))),
bvco-bitvec (carry-col2 (bv-vec (bv-vec (bv-vec (ik)))),
                                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))),
                                bvco-carry (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))))),

```

```

                                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))

```

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

DEFINITION:

$$\begin{aligned}
& \text{induct-carry-col2}(ik, ip, ccin, nb) \\
= & \text{if bitvecp}(ik) \\
& \quad \wedge \text{bitvecp}(ip) \\
& \quad \wedge \text{boolp}(ccin) \\
& \quad \wedge (\text{bv-size}(ik) = \text{bv-size}(ip)) \\
& \quad \wedge \text{evenp}(nb) \\
& \text{then if } ik = \text{BV-NIL} \text{ then t} \\
& \quad \text{elseif}(\text{bv-size}(ik) < \text{up}(nb)) \\
& \quad \quad \wedge (11 < nb) \\
& \quad \quad \wedge (4 < \text{bv-size}(ik)) \\
& \quad \quad \wedge \text{evenp}(\text{bv-size}(ik)) \\
& \quad \text{then induct-carry-col2}(\text{bv-vec}(\text{bv-vec}(\text{bv-vec}(\text{bv-vec}(ik))), \\
& \quad \quad \text{bv-vec}(\text{bv-vec}(\text{bv-vec}(\text{bv-vec}(ip))), \\
& \quad \quad \text{ccin}, \\
& \quad \quad nb) \\
& \quad \quad \wedge \text{induct-carry-col2}(\text{bv-vec}(ik), \text{bv-vec}(ip), ccin, nb) \\
& \quad \text{else induct-carry-col2}(\text{bv-vec}(ik), \text{bv-vec}(ip), ccin, nb) \text{ endif} \\
& \text{else t endif}
\end{aligned}$$

THEOREM: carry2-carry-relate

$\text{evenp}(nb) \rightarrow (\text{carry-col2}(ik, ip, ccin, nb) = \text{carry-col}(ik, ip, ccin))$

DEFINITION:

$\text{cbyp-col}(ik, ip, ccin, nb)$

= **if**  $\text{bitvecp}(ik)$

$\wedge$   $\text{bitvecp}(ip)$

$\wedge$   $\text{boolp}(ccin)$

$\wedge$   $(\text{bv-size}(ik) = \text{bv-size}(ip))$

$\wedge$   $\text{evenp}(nb)$

**then if**  $ik = \text{BV-NIL}$  **then**  $\text{bitvec-carry-ovf}(\text{BV-NIL}, ccin, \mathbf{f})$

**elseif**  $(\text{bv-size}(ik) < \text{up}(nb))$

$\wedge$   $(11 < nb)$

$\wedge$   $(4 < \text{bv-size}(ik))$

$\wedge$   $\text{evenp}(\text{bv-size}(ik))$

**then**  $\text{bitvec-carry-ovf}(\text{bv-append}(\text{bvco-bitvec}(\text{c4byp}(\text{bv-bit}(ip),$

$\text{bv-bit}(\text{bv-vec}(ip)),$

$\text{bv-bit}(\text{bv-vec}(\text{bv-vec}(ip))),$

$\text{bv-bit}(\text{bv-vec}(\text{bv-vec}(\text{bv-vec}(ip))))),$

$\text{bv-bit}(ik),$

$\text{bv-bit}(\text{bv-vec}(ik)),$

$\text{bv-bit}(\text{bv-vec}(\text{bv-vec}(ik))),$

$\text{bv-bit}(\text{bv-vec}(\text{bv-vec}(\text{bv-vec}(ik))))),$

$\text{bvco-carry}(\text{cbyp-col}(\text{bv-vec}(\text{bv-vec}(\text{bv-vec}(\text{bv-vec}(ip),$

$\text{bv-vec}(\text{bv-vec}(\text{bv-vec}(ip))),$

$ccin,$

$nb))))),$

$\text{bvco-bitvec}(\text{cbyp-col}(\text{bv-vec}(\text{bv-vec}(\text{bv-vec}(\text{bv-vec}(ik))),$

$\text{bv-vec}(\text{bv-vec}(\text{bv-vec}(ip))),$

$ccin,$

$nb))))),$

$\text{bvco-carry}(\text{c4byp}(\text{bv-bit}(ip),$

$\text{bv-bit}(\text{bv-vec}(ip)),$

$\text{bv-bit}(\text{bv-vec}(\text{bv-vec}(ip))),$

$\text{bv-bit}(\text{bv-vec}(\text{bv-vec}(\text{bv-vec}(ip))))),$

$\text{bv-bit}(ik),$

$\text{bv-bit}(\text{bv-vec}(ik)),$

$\text{bv-bit}(\text{bv-vec}(\text{bv-vec}(ik))),$

$\text{bv-bit}(\text{bv-vec}(\text{bv-vec}(\text{bv-vec}(ik))))),$

$\text{bvco-carry}(\text{cbyp-col}(\text{bv-vec}(\text{bv-vec}(\text{bv-vec}(\text{bv-vec}(ik))),$

$\text{bv-vec}(\text{bv-vec}(\text{bv-vec}(ip))),$

$ccin,$

$nb))))),$

$\text{bvco-carry}(\text{cbyp-col}(\text{bv-vec}(ik),$

$\text{bv-vec}(ip),$

```

                                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

$$\text{cbyp-col}(ik, ip, ccin, nb) = \text{carry-col2}(ik, ip, ccin, nb)$$

THEOREM: cbyp-carry-relate

$$\text{evenp}(nb) \rightarrow (\text{cbyp-col}(k, p, cin, nb) = \text{carry-col}(k, p, cin))$$

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**  $\neg$  *ct1*  
**else**  $\neg$  *ct0* **endif**

DEFINITION:

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

THEOREM: alugen-alugenod-relate

alugen-o(*ct3*, *ct2*, *ct1*, *ct0*, *a*, *b*) = alugenod-o(*ct3*, *ct2*, *ct1*, *ct0*, *a*,  $\neg$  *b*)

THEOREM: alugenodo-func

(boolp(*a*)  $\wedge$  boolp(*b*))

$\rightarrow$  ((alugenod-o(**t**, **t**, **t**, **t**, *a*, *b*) = **f**)  
 $\wedge$  (alugenod-o(**t**, **t**, **t**, **f**, *a*, *b*) = (*a*  $\wedge$  ( $\neg$  *b*)))  
 $\wedge$  (alugenod-o(**t**, **t**, **f**, **t**, *a*, *b*) = (*a*  $\wedge$  *b*))  
 $\wedge$  (alugenod-o(**t**, **t**, **f**, **f**, *a*, *b*) = *a*)  
 $\wedge$  (alugenod-o(**t**, **f**, **t**, **t**, *a*, *b*) = (( $\neg$  *a*)  $\wedge$  *b*))  
 $\wedge$  (alugenod-o(**t**, **f**, **t**, **f**, *a*, *b*) = exor(*a*, *b*))  
 $\wedge$  (alugenod-o(**t**, **f**, **f**, **t**, *a*, *b*) = *b*)  
 $\wedge$  (alugenod-o(**t**, **f**, **f**, **f**, *a*, *b*) = (*a*  $\vee$  *b*))  
 $\wedge$  (alugenod-o(**f**, **t**, **t**, **t**, *a*, *b*) = ( $\neg$  (*a*  $\vee$  *b*)))  
 $\wedge$  (alugenod-o(**f**, **t**, **t**, **f**, *a*, *b*) = ( $\neg$  *b*))  
 $\wedge$  (alugenod-o(**f**, **t**, **f**, **t**, *a*, *b*) = ( $\neg$  exor(*a*, *b*)))  
 $\wedge$  (alugenod-o(**f**, **t**, **f**, **f**, *a*, *b*) = (*a*  $\vee$  ( $\neg$  *b*)))  
 $\wedge$  (alugenod-o(**f**, **f**, **t**, **t**, *a*, *b*) = ( $\neg$  *a*))  
 $\wedge$  (alugenod-o(**f**, **f**, **t**, **f**, *a*, *b*) = ( $\neg$  (*a*  $\wedge$  *b*)))  
 $\wedge$  (alugenod-o(**f**, **f**, **f**, **t**, *a*, *b*) = (( $\neg$  *a*)  $\vee$  *b*))  
 $\wedge$  (alugenod-o(**f**, **f**, **f**, **f**, *a*, *b*) = **t**))

DEFINITION: cscbi11(*input*) = ( $\neg$  *input*)

DEFINITION: cscbo11(*input*) = ( $\neg$  *input*)

DEFINITION:

prop-col(*ina*, *inb*, *cp*)

= **if** bitvecp(*ina*)  
 $\wedge$  bitvecp(*inb*)  
 $\wedge$  controlep(*cp*)  
 $\wedge$  (bv-size(*ina*) = bv-size(*inb*))  
**then if** *ina* = BV-NIL **then** BV-NIL  
**else** bitvec(alugen-o(cscbi11(bv-3(*cp*))),

```

                                cscbil1 (bv-2 (cp)),
                                cscbil1 (bv-1 (cp)),
                                cscbil1 (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  
 $(\text{controlep}(cp) \wedge \text{bitvecp}(a) \wedge \text{bitvecp}(b) \wedge (\text{bv-size}(a) = \text{bv-size}(b)))$   
 $\rightarrow (\text{bv-size}(\text{prop-col}(a, b, cp)) = \text{bv-size}(a))$

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

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

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

DEFINITION:  
 $\text{kill-col}(ina, inb, ck)$   
 $=$  **if**  $\text{bitvecp}(ina)$   
 $\quad \wedge \text{bitvecp}(inb)$   
 $\quad \wedge \text{controlep}(ck)$   
 $\quad \wedge (\text{bv-size}(ina) = \text{bv-size}(inb))$   
**then if**  $ina = \text{BV-NIL}$  **then**  $\text{BV-NIL}$   
 $\quad \text{else bitvec}(\text{alugen-o}(\text{cscbil1}(\text{bv-3}(ck)),$   
 $\quad \quad \text{cscbil1}(\text{bv-2}(ck)),$   
 $\quad \quad \text{cscbil1}(\text{bv-1}(ck)),$   
 $\quad \quad \text{cscbil1}(\text{bv-0}(ck)),$   
 $\quad \quad \text{bv-bit}(ina),$   
 $\quad \quad \text{bv-bit}(inb)),$   
 $\quad \text{kill-col}(\text{bv-vec}(ina), \text{bv-vec}(inb), ck))$  **endif**  
**else BV-NIL endif**

THEOREM: prop-kill-relate  
 $\text{kill-col}(a, b, control) = \text{prop-col}(a, b, control)$

THEOREM: size-kill-col  
 $(\text{controlep}(ck) \wedge \text{bitvecp}(a) \wedge \text{bitvecp}(b) \wedge (\text{bv-size}(a) = \text{bv-size}(b)))$   
 $\rightarrow (\text{bv-size}(\text{kill-col}(a, b, ck)) = \text{bv-size}(a))$



THEOREM: kill-col-12

$$\begin{aligned} & (\text{bitvecp}(a) \wedge \text{bitvecp}(b) \wedge (\text{bv-size}(a) = \text{bv-size}(b))) \\ \rightarrow & (\text{kill-col}(a, b, \text{nat-to-bv}(12, 4)) = \text{bv-not}(a)) \end{aligned}$$

EVENT: Disable prop-kill-relate.

DEFINITION:

$\text{res-col}(ina, inb, cr)$

```
= if bitvecp(ina)
   $\wedge$  bitvecp(inb)
   $\wedge$  controlep(cr)
   $\wedge$  (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

$$\begin{aligned} & (\text{controlep}(cr) \wedge \text{bitvecp}(a) \wedge \text{bitvecp}(b) \wedge (\text{bv-size}(a) = \text{bv-size}(b))) \\ \rightarrow & (\text{bv-size}(\text{res-col}(a, b, cr)) = \text{bv-size}(a)) \end{aligned}$$

THEOREM: res-col-10

$$\begin{aligned} & (\text{bitvecp}(a) \wedge \text{bitvecp}(b) \wedge (\text{bv-size}(a) = \text{bv-size}(b))) \\ \rightarrow & (\text{res-col}(a, b, \text{nat-to-bv}(10, 4)) = \text{bv-exor}(a, \text{bv-invert-even}(b))) \end{aligned}$$

THEOREM: lemma1

$$\begin{aligned} & (\text{bitvecp}(a) \\ & \wedge \text{bitvecp}(b) \\ & \wedge (\text{bv-size}(a) = \text{bv-size}(b)) \\ & \wedge \text{controlep}(cp) \\ & \wedge \text{controlep}(ck) \\ & \wedge \text{boolp}(ccin)) \end{aligned}$$

→ (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:  $\text{csexor}(a, b) = \text{exor}(a, b)$

DEFINITION:

$\text{mcalu}(nb, byp, ina, inb, ck, cp, cr, ccin)$

= **if**  $\text{evenp}(nb)$

$\wedge$   $\text{bitvecp}(ina)$

$\wedge$   $\text{bitvecp}(inb)$

$\wedge$   $\text{boolp}(ccin)$

$\wedge$   $\text{boolp}(byp)$

$\wedge$   $\text{controlep}(ck)$

$\wedge$   $\text{controlep}(cp)$

$\wedge$   $\text{controlep}(cr)$

$\wedge$   $(\text{bv-size}(ina) = \text{bv-size}(inb))$

**then**  $\text{carry-sign-ovf-bitvec}(\text{cscbo11}(\text{bvco-carry}(\text{rec-mcalu-imp}(nb,$

$byp \wedge (11 < nb),$

$ina,$

$inb,$

$ck,$

$cp,$

$cr,$

$\text{cscbi11}(ccin))))),$

$\text{cscbo11}(\text{bv-bit}(\text{bvco-bitvec}(\text{rec-mcalu-imp}(nb,$

$byp \wedge (11 < nb),$

$ina,$

$inb,$

$ck,$

$cp,$

$cr,$

$\text{cscbi11}(ccin))))),$

$\text{csexor}(\text{bvco-carry}(\text{rec-mcalu-imp}(nb,$

$byp \wedge (11 < nb),$

$ina,$

$inb,$

```

                                ck,
                                cp,
                                cr,
                                cscbill(ccin)),
bvco-ovf(rec-mcalu-imp(nb,
                        byp ∧ (11 < nb),
                        ina,
                        inb,
                        ck,
                        cp,
                        cr,
                        cscbill(ccin))),
bvco-bitvec(rec-mcalu-imp(nb,
                          byp ∧ (11 < nb),
                          ina,
                          inb,
                          ck,
                          cp,
                          cr,
                          cscbill(ccin)))

```

**else CSOBV-NIL endif**

DEFINITION:

induct-vec-vec-evenp-f( $a, b, c$ )

```

= if bitvecp( $a$ )
  ∧ bitvecp( $b$ )
  ∧ (bv-size( $a$ ) = bv-size( $b$ ))
  ∧ 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( $a$ ) ∧ bitvecp( $b$ ) ∧ (bv-size( $a$ ) = bv-size( $b$ )) ∧ boolp( $cin$ ))
→ (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 ¬ bvco-carry(bv-adder( $a, b, cin$ )) endif)

```

THEOREM: the-proof-part2

```

(bitvecp( $a$ ) ∧ bitvecp( $b$ ) ∧ (bv-size( $a$ ) = bv-size( $b$ )) ∧ boolp( $cin$ ))
→ (bv-exor(bv-exor( $a, b$ ),
              bv-invert-even(bvco-bitvec(carry-col(bv-not( $a$ ),
                                                    bv-exor( $a, b$ ),

```

$$= \text{bvco-bitvec}(\text{bv-adder}(a, b, \text{cin})))$$

THEOREM: the-proof-part3

$$\begin{aligned} & (\text{bitvecp}(a) \wedge \text{bitvecp}(b) \wedge (\text{bv-size}(a) = \text{bv-size}(b)) \wedge \text{boolp}(\text{cin})) \\ \rightarrow & (\text{bvco-ovf}(\text{carry-col}(\text{bv-not}(a), \text{bv-exor}(a, b), \text{cin})) \\ & = \text{if evenp}(\text{bv-size}(a)) \\ & \quad \text{then if } a = \text{BV-NIL} \text{ then f} \\ & \quad \quad \text{else } \neg \text{bvco-ovf}(\text{bv-adder}(a, b, \text{cin})) \text{ endif} \\ & \quad \text{else } \text{bvco-ovf}(\text{bv-adder}(a, b, \text{cin})) \text{ endif} \end{aligned}$$

THEOREM: rec-mcalu-imp-bv-adder-relate

$$\begin{aligned} & (\text{bitvecp}(a) \\ & \wedge \text{bitvecp}(b) \\ & \wedge \text{boolp}(\text{cin}) \\ & \wedge \text{boolp}(\text{byp}) \\ & \wedge (\text{bv-size}(a) = \text{bv-size}(b)) \\ & \wedge \text{evenp}(nb) \\ & \wedge (\text{bv-size}(a) = nb)) \\ \rightarrow & (\text{bvco-bitvec}(\text{rec-mcalu-imp}(nb, \\ & \quad \text{byp}, \\ & \quad a, \\ & \quad b, \\ & \quad \text{nat-to-bv}(12, 4), \\ & \quad \text{nat-to-bv}(10, 4), \\ & \quad \text{nat-to-bv}(10, 4), \\ & \quad \text{cin})) \\ & = \text{bvco-bitvec}(\text{bv-adder}(a, b, \text{cin}))) \end{aligned}$$

THEOREM: mcalu-imp-bv-adder-relate

$$\begin{aligned} & (\text{bitvecp}(a) \\ & \wedge \text{bitvecp}(b) \\ & \wedge \text{boolp}(\text{cin}) \\ & \wedge \text{boolp}(\text{byp}) \\ & \wedge (\text{bv-size}(a) = \text{bv-size}(b)) \\ & \wedge \text{evenp}(nb) \\ & \wedge (\text{bv-size}(a) = nb)) \\ \rightarrow & (\text{csobv-bitvec}(\text{mcalu}(nb, \\ & \quad \text{byp}, \\ & \quad a, \\ & \quad b, \\ & \quad \text{nat-to-bv}(12, 4), \\ & \quad \text{nat-to-bv}(10, 4), \\ & \quad \text{nat-to-bv}(10, 4), \\ & \quad \text{cin})) \end{aligned}$$

$$= \text{bvco-bitvec}(\text{bv-adder}(a, b, \neg \text{cin}))$$

THEOREM: boolp-not  
 $\text{boolp}(\neg a)$

THEOREM: tc-interpretation-of-mcalu-output  
 $(\text{bitvecp}(a)$

$$\begin{aligned} & \wedge \text{bitvecp}(b) \\ & \wedge (\text{bv-size}(a) = \text{bv-size}(b)) \\ & \wedge \text{boolp}(\text{cin}) \\ & \wedge \text{boolp}(\text{byp}) \\ & \wedge \text{evenp}(nb) \\ & \wedge (\text{bv-size}(a) = nb) \\ & \wedge (a \neq \text{BV-NIL}) \\ & \wedge (b \neq \text{BV-NIL}) \\ \rightarrow & (\text{tc-to-integer}(\text{csobv-bitvec}(\text{mcalu}(nb, \\ & \qquad \qquad \qquad \text{byp}, \\ & \qquad \qquad \qquad a, \\ & \qquad \qquad \qquad b, \\ & \qquad \qquad \text{nat-to-bv}(12, 4), \\ & \qquad \qquad \text{nat-to-bv}(10, 4), \\ & \qquad \qquad \text{nat-to-bv}(10, 4), \\ & \qquad \qquad \text{cin}))) \\ = & \text{ if tc-in-rangep}(\text{add}(\text{tc-to-integer}(a), \\ & \qquad \qquad \qquad \text{add}(\text{tc-to-integer}(b), \text{carry}(\neg \text{cin}))), \\ & \qquad \qquad \qquad \text{bv-size}(a)) \\ & \text{ then } \text{add}(\text{tc-to-integer}(a), \\ & \qquad \qquad \qquad \text{add}(\text{tc-to-integer}(b), \text{carry}(\neg \text{cin}))) \\ & \text{ elseif } \text{negativep}(\text{add}(\text{tc-to-integer}(a), \\ & \qquad \qquad \qquad \text{add}(\text{tc-to-integer}(b), \text{carry}(\neg \text{cin})))) \\ & \text{ then } \text{add}(\text{tc-to-integer}(a), \\ & \qquad \qquad \text{add}(\text{tc-to-integer}(b), \\ & \qquad \qquad \text{add}(\text{carry}(\neg \text{cin}), \text{twoto}(\text{bv-size}(a)))))) \\ & \text{ else } \text{add}(\text{tc-to-integer}(a), \\ & \qquad \qquad \text{add}(\text{tc-to-integer}(b), \\ & \qquad \qquad \text{add}(\text{carry}(\neg \text{cin}), - \text{twoto}(\text{bv-size}(a)))))) \text{ endif} \end{aligned}$$

THEOREM: bv-to-nat-of-bv-not

$$\text{bv-to-nat}(\text{bv-not}(a)) = ((\text{twoto}(\text{bv-size}(a)) - 1) - \text{bv-to-nat}(a))$$

DEFINITION:

$$\begin{aligned} & \text{tc-minus}(a) \\ = & \text{ if } \text{negativep}(a) \text{ then } \text{negative-guts}(a) \\ & \text{ elseif } a \simeq 0 \text{ then } 0 \\ & \text{ else } -a \text{ endif} \end{aligned}$$

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

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

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

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

DEFINITION:  
 $\text{tc-fix}(x)$   
 $= \text{if } \text{tcp}(x) \text{ then } x$   
 $\text{else } 0 \text{ endif}$

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

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

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  
 $(\text{bitvecp}(a) \wedge (a \neq \text{BV-NIL})) \rightarrow (\text{bv-not}(a) \neq \text{BV-NIL})$

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

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

THEOREM: tcp-tc-minus  
 $\text{tcp}(\text{tc-minus}(a))$

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

THEOREM: tcp-twoto  
 $\text{tcp}(\text{twoto}(a))$

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

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



```

                                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 ≠ BV-NIL)
  ^ (b ≠ 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)
  ^ bitvecp (b)
  ^ (bv-size (a) = bv-size (b))
  ^ (a ≠ BV-NIL)
  ^ (b ≠ BV-NIL)
  ^ 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),
                                     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)
  ^ bitvecp (b)
  ^ (bv-size (a) = bv-size (b))
  ^ boolp (cin)
  ^ (a ≠ BV-NIL)
→ ((bvco-bitvec (bv-adder (a, b, cin)) = BV-NIL) = f)

```

DEFINITION:

```

last (it)
= if bitvecp (it) ^ (it ≠ BV-NIL)

```

**then if**  $\text{bv-vec}(it) = \text{BV-NIL}$  **then**  $\text{bv-bit}(it)$   
     **else**  $\text{last}(\text{bv-vec}(it))$  **endif**  
**else f endif**

DEFINITION:

$\text{butlast}(it)$   
 $=$  **if**  $\text{bitvecp}(it) \wedge (it \neq \text{BV-NIL})$   
     **then if**  $\text{bv-vec}(it) = \text{BV-NIL}$  **then**  $\text{BV-NIL}$   
         **else**  $\text{bitvec}(\text{bv-bit}(it), \text{butlast}(\text{bv-vec}(it)))$  **endif**  
     **else**  $\text{BV-NIL}$  **endif**

DEFINITION:

$\text{do-shift-1}(it, \text{shiftin})$   
 $=$  **if**  $\text{bitvecp}(it) \wedge (it \neq \text{BV-NIL}) \wedge \text{boolp}(\text{shiftin})$   
     **then**  $\text{bitvec}(\text{shiftin}, \text{butlast}(it))$   
     **else**  $\text{BV-NIL}$  **endif**

DEFINITION:

$\text{up-shift-1}(it, \text{shiftin})$   
 $=$  **if**  $\text{bitvecp}(it) \wedge (it \neq \text{BV-NIL}) \wedge \text{boolp}(\text{shiftin})$   
     **then**  $\text{bv-append}(\text{bv-vec}(it), \text{bitvec}(\text{shiftin}, \text{BV-NIL}))$   
     **else**  $\text{BV-NIL}$  **endif**

DEFINITION:

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

DEFINITION:

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

THEOREM: size-bv-append

$(\text{bitvecp}(a) \wedge \text{bitvecp}(b))$   
 $\rightarrow (\text{bv-size}(\text{bv-append}(a, b)) = (\text{bv-size}(a) + \text{bv-size}(b)))$

THEOREM: append-not-nil

$(\text{bitvecp}(a) \wedge \text{bitvecp}(b) \wedge (a \neq \text{BV-NIL}))$   
 $\rightarrow ((\text{bv-append}(a, b) = \text{BV-NIL}) = \mathbf{f})$

THEOREM: vec-append

$$\begin{aligned} & ((a \neq \text{BV-NIL}) \wedge \text{bitvecp}(a) \wedge \text{bitvecp}(b)) \\ & \rightarrow (\text{bv-vec}(\text{bv-append}(a, b)) = \text{bv-append}(\text{bv-vec}(a), b)) \end{aligned}$$

THEOREM: size-up-shift-1

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

THEOREM: size-up-shift-n

$$\begin{aligned} & (\text{boolp}(\text{shiftin}) \wedge (n \in \mathbf{N})) \\ & \rightarrow (\text{bv-size}(\text{up-shift-n}(n, it, \text{shiftin})) = \text{bv-size}(it)) \end{aligned}$$

THEOREM: size-do-shift-n

$$\begin{aligned} & (\text{boolp}(\text{shiftin}) \wedge (n \in \mathbf{N})) \\ & \rightarrow (\text{bv-size}(\text{do-shift-n}(n, it, \text{shiftin})) = \text{bv-size}(it)) \end{aligned}$$

THEOREM: bv-append-not-not

$$\begin{aligned} & (\text{bitvecp}(a) \wedge \text{bitvecp}(b)) \\ & \rightarrow (\text{bv-not}(\text{bv-append}(a, b)) = \text{bv-append}(\text{bv-not}(a), \text{bv-not}(b))) \end{aligned}$$

THEOREM: up-shift-0

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

THEOREM: hack2

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

THEOREM: evenp-twoto

$$(a \neq 0) \rightarrow \text{evenp}(\text{twoto}(a))$$

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

$$\begin{aligned} & ((a \in \mathbf{N}) \wedge (b \in \mathbf{N})) \\ & \rightarrow (\text{evenp}(a + b) \\ & \quad = \text{if evenp}(a) \text{ then evenp}(b) \\ & \quad \quad \text{else } \neg \text{evenp}(b) \text{ endif}) \end{aligned}$$

THEOREM: do-shift-strict

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

THEOREM: do-shift-0

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

DEFINITION:

$$\begin{aligned} & \text{mult}(a, b) \\ & = \text{if negativep}(a) \\ & \quad \text{then if negativep}(b) \text{ then negative-guts}(a) * \text{negative-guts}(b) \\ & \quad \quad \text{else tc-minus}(\text{negative-guts}(a) * b) \text{ endif} \\ & \quad \text{elseif negativep}(b) \text{ then tc-minus}(a * \text{negative-guts}(b)) \\ & \quad \text{else } a * b \text{ endif} \end{aligned}$$

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)  $\wedge$  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  $\not\prec$  b)  $\rightarrow$  (((a - b) **mod** b) = (a **mod** b))

THEOREM: remainder-diff-times  
(x  $\not\prec$  (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 ( $it$ )  $\wedge$  ( $it \neq \text{BV-NIL}$ ))
 $\rightarrow$  (tc-to-integer (up-shift-1 ( $it$ ,  $\mathbf{f}$ ))
    = 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

$$\begin{aligned} & ((z < 2) \wedge \text{evenp}(z + (2 * a))) \\ \rightarrow & ((z + (2 * a)) = (2 * a)) \end{aligned}$$

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: multiplep( $a, n$ ) =  $((a \bmod n) \simeq 0)$

THEOREM: negative-guts-tc-minus

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

DEFINITION:

$$\text{invert-lsb}(it) = \text{bv-append}(\text{butlast}(it), \text{bitvec}(\neg \text{last}(it), \text{BV-NIL}))$$

THEOREM: lsb-implies-odd

$$\text{last}(it) = (\neg \text{evenp}(\text{bv-to-nat}(it)))$$

THEOREM: bv-bit-of-bv-append

$$\begin{aligned} & (\text{bitvecp}(a) \wedge (a \neq \text{BV-NIL}) \wedge \text{bitvecp}(b)) \\ \rightarrow & (\text{bv-bit}(\text{bv-append}(a, b)) = \text{bv-bit}(a)) \end{aligned}$$

THEOREM: butlast-bv-append

$$\begin{aligned} & (\text{bitvecp}(b) \wedge (b \neq \text{BV-NIL})) \\ \rightarrow & (\text{butlast}(\text{bv-append}(a, b)) = \text{bv-append}(a, \text{butlast}(b))) \end{aligned}$$

THEOREM: bv-append-of-bv-nil

$$\text{bitvecp}(a) \rightarrow (\text{bv-append}(a, \text{BV-NIL}) = a)$$

THEOREM: up-shift-1-invert-lsb

$$\begin{aligned} & (\text{bitvecp}(it) \wedge (it \neq \text{BV-NIL})) \\ \rightarrow & (\text{up-shift-1}(it, \mathbf{t}) = \text{invert-lsb}(\text{up-shift-1}(it, \mathbf{f}))) \end{aligned}$$

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

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

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

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

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

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

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

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

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



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

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

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

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

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

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

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

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

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

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

THEOREM: multiplep-tc-minus

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

THEOREM: imultiplep-new-r

$$\begin{aligned} & (\text{tcp}(r) \wedge \text{tcp}(n) \wedge (i \in \mathbf{N}) \wedge \text{imultiplep}(r, n)) \\ \rightarrow & \text{imultiplep}(\text{new-r}(i, r, n), n) \end{aligned}$$

THEOREM: remainder-diff-times-commuted

$$\begin{aligned} & ((a \in \mathbf{N}) \wedge (x \in \mathbf{N}) \wedge (b \in \mathbf{N}) \wedge (x < b) \wedge ((a * b) \not\leq x)) \\ \rightarrow & (((a * b) - x) \bmod b) \\ & = \text{if } x = 0 \text{ then } 0 \\ & \quad \text{else } b - x \text{ endif} \end{aligned}$$

DEFINITION:

times-times-induct( $a, b, c$ )

$$\begin{aligned} = & \text{if } b \simeq 0 \text{ then } t \\ & \text{else times-times-induct}(a, b - 1, c - 1) \text{ endif} \end{aligned}$$

THEOREM: not-equal-times-0

$$\begin{aligned} & ((a \in \mathbf{N}) \wedge (b \in \mathbf{N}) \wedge (a \neq 0) \wedge (b \neq 0)) \\ \rightarrow & ((0 * a) \neq (a * b)) \end{aligned}$$

THEOREM: equal-times-times

$$\begin{aligned} & ((a \in \mathbf{N}) \wedge (b \in \mathbf{N}) \wedge (c \in \mathbf{N}) \wedge (a \neq 0)) \\ \rightarrow & (((a * b) = (a * c)) = (b = c)) \end{aligned}$$

THEOREM: hack-around-next-lemma

$$\begin{aligned} & ((w \in \mathbf{N}) \\ & \wedge (z \in \mathbf{N}) \\ & \wedge (z < b) \\ & \wedge (v \in \mathbf{N}) \\ & \wedge (b \neq 0) \\ & \wedge (b \in \mathbf{N}) \\ & \wedge (w \neq 0) \\ & \wedge (z \neq 0) \\ & \wedge ((b + (b * w)) \not\leq (z + (b * v))) \\ & \wedge ((b * w) < (z + (b * v)))) \\ \rightarrow & (((1 + v) - 1) = w) \end{aligned}$$

THEOREM: lower-upper-determines

$$\begin{aligned} & ((a \in \mathbf{N}) \\ & \wedge (b \in \mathbf{N}) \\ & \wedge (a \neq 0) \\ & \wedge (a \neq 1) \\ & \wedge (x \in \mathbf{N}) \end{aligned}$$

$\wedge (x \neq 0)$   
 $\wedge ((x \bmod b) \neq 0)$   
 $\wedge ((a * b) \not\leq x)$   
 $\wedge (((a - 1) * b) < x)$   
 $\rightarrow ((1 + (x \div b)) = a)$

THEOREM: remainder-diff-times-commuted-2

$((a \in \mathbf{N}) \wedge (x \in \mathbf{N}) \wedge (b \in \mathbf{N}) \wedge (x \not\leq b) \wedge ((a * b) \not\leq x))$   
 $\rightarrow (((a * b) - x) \bmod b)$   
 $= \text{if } (x \bmod b) = 0 \text{ then } 0$   
 $\text{else } b - (x \bmod b) \text{ endif}$

THEOREM: not-imultiplep-add

$(\text{tcp}(a) \wedge \text{tcp}(b) \wedge (x \in \mathbf{N}) \wedge (\neg \text{imultiplep}(a, b)))$   
 $\rightarrow (\neg \text{imultiplep}(\text{add}(a, \text{mult}(b, \text{twoto}(x))), b))$

THEOREM: not-imultiplep-new-r

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

THEOREM: unfold-positive-tc-to-integer

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

THEOREM: unfold-negative-tc-to-integer

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

DEFINITION:

$\text{div-in-rangep}(tel, noem, i)$

$= \text{if } tel \in \mathbf{N}$   
 $\text{then if } noem \in \mathbf{N} \text{ then } tel < (\text{twoto}(i - 1) * noem)$   
 $\text{else } tel < (\text{twoto}(i - 1) * \text{negative-guts}(noem)) \text{ endif}$   
 $\text{elseif } noem \in \mathbf{N} \text{ then } (\text{twoto}(i - 1) * noem) \not\leq \text{negative-guts}(tel)$   
 $\text{else } (\text{twoto}(i - 1) * \text{negative-guts}(noem))$   
 $\not\leq \text{negative-guts}(tel) \text{ endif}$

THEOREM: div-in-rangep-new-r-sign-1

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

THEOREM: div-in-rangep-new-r-sign-2

$(\text{div-in-rangep}(tel, noem, i) \wedge (tel \in \mathbf{N}) \wedge \text{negativep}(noem))$   
 $\rightarrow \text{negativep}(\text{add}(tel, \text{mult}(noem, \text{twoto}(i - 1))))$

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

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

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

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

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

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

EVENT: Disable quotient-times-lessp.

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

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

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

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

$$\begin{aligned}
& \wedge ((\text{negative-guts}(x) \bmod y) \neq 0) \\
& \wedge ((y * z) \not\prec \text{negative-guts}(x)) \\
& \wedge ((\text{negative-guts}(x) \div y) = 0) \\
& \wedge (z < 1) \\
& \rightarrow (\text{divide}((y * z) - \text{negative-guts}(x), y) = \text{add}(- (1 - z), 0))
\end{aligned}$$

THEOREM: may-be-baby-4

$$\begin{aligned}
& ((x \in \mathbf{N}) \\
& \wedge (y \in \mathbf{N}) \\
& \wedge (z \in \mathbf{N}) \\
& \wedge (z \not\prec 1) \\
& \wedge (x \neq 0) \\
& \wedge (y \neq 0) \\
& \wedge ((y * z) \not\prec x) \\
& \wedge ((x \bmod y) \neq 0)) \\
& \rightarrow ((z - 1) \not\prec (x \div y))
\end{aligned}$$

THEOREM: may-be-baby-5

$$\begin{aligned}
& ((x \in \mathbf{N}) \\
& \wedge (x \neq 0) \\
& \wedge (y \in \mathbf{N}) \\
& \wedge (y \neq 0) \\
& \wedge (y \not\prec x) \\
& \wedge ((x \bmod y) \neq 0)) \\
& \rightarrow ((x \div y) = 0)
\end{aligned}$$

THEOREM: may-be-baby-6

$$\begin{aligned}
& (\text{negativep}(x) \\
& \wedge (\text{negative-guts}(x) \neq 0) \\
& \wedge \text{negativep}(y) \\
& \wedge (\text{negative-guts}(y) \neq 0) \\
& \wedge \text{negativep}(z) \\
& \wedge (\text{negative-guts}(z) \neq 0) \\
& \wedge ((\text{negative-guts}(x) \bmod \text{negative-guts}(y)) \neq 0) \\
& \wedge ((\text{negative-guts}(y) * \text{negative-guts}(z)) \not\prec \text{negative-guts}(x)) \\
& \wedge (1 < \text{negative-guts}(z))) \\
& \rightarrow (\text{divide}((\text{negative-guts}(y) * \text{negative-guts}(z)) - \text{negative-guts}(x), \\
& \quad y) \\
& \quad = \text{add}(\text{negative-guts}(x) \div \text{negative-guts}(y), \\
& \quad \quad - (\text{negative-guts}(z) - 1)))
\end{aligned}$$

THEOREM: equal-times-times-commuted

$$\begin{aligned}
& ((a \in \mathbf{N}) \wedge (b \in \mathbf{N}) \wedge (c \in \mathbf{N}) \wedge (a \neq 0)) \\
& \rightarrow (((a * b) = (c * a)) = (b = c))
\end{aligned}$$

THEOREM: divide-add-mult

$$\begin{aligned} & (\text{tcp}(x) \wedge \text{tcp}(y) \wedge \text{tcp}(z)) \\ \rightarrow & (\text{divide}(\text{add}(x, \text{mult}(y, z)), y) \\ = & \text{if } y = 0 \text{ then } 0 \\ & \text{elseif imultiplep}(x, y) \text{ then add}(\text{divide}(x, y), z) \\ & \text{elseif } x \in \mathbf{N} \\ & \text{then if add}(x, \text{mult}(y, z)) \in \mathbf{N} \text{ then add}(\text{divide}(x, y), z) \\ & \quad \text{elseif } y \in \mathbf{N} \text{ then add}(\text{divide}(x, y), \text{add}(z, 1)) \\ & \quad \text{else add}(\text{divide}(x, y), \text{add}(z, -1)) \text{ endif} \\ & \text{elseif negativep}(\text{add}(x, \text{mult}(y, z))) \text{ then add}(\text{divide}(x, y), z) \\ & \text{elseif } y \in \mathbf{N} \text{ then add}(\text{divide}(x, y), \text{add}(z, -1)) \\ & \text{else add}(\text{divide}(x, y), \text{add}(z, 1)) \text{ endif} \end{aligned}$$

THEOREM: div-in-range-sign-new-r-1

$$\begin{aligned} & ((\text{tel} \in \mathbf{N}) \wedge (\text{noem} \in \mathbf{N}) \wedge \text{div-in-rangep}(\text{tel}, \text{noem}, i)) \\ \rightarrow & \text{negativep}(\text{new-r}(i, \text{tel}, \text{noem})) \end{aligned}$$

THEOREM: div-in-range-sign-new-r-2

$$\begin{aligned} & ((\text{tel} \in \mathbf{N}) \wedge \text{negativep}(\text{noem}) \wedge \text{div-in-rangep}(\text{tel}, \text{noem}, i)) \\ \rightarrow & \text{negativep}(\text{new-r}(i, \text{tel}, \text{noem})) \end{aligned}$$

THEOREM: div-in-range-sign-new-r-3

$$\begin{aligned} & (\text{negativep}(\text{tel}) \wedge (\text{noem} \in \mathbf{N}) \wedge \text{div-in-rangep}(\text{tel}, \text{noem}, i)) \\ \rightarrow & (\text{new-r}(i, \text{tel}, \text{noem}) \in \mathbf{N}) \end{aligned}$$

THEOREM: div-in-range-sign-new-r-4

$$\begin{aligned} & (\text{negativep}(\text{tel}) \wedge \text{negativep}(\text{noem}) \wedge \text{div-in-rangep}(\text{tel}, \text{noem}, i)) \\ \rightarrow & (\text{new-r}(i, \text{tel}, \text{noem}) \in \mathbf{N}) \end{aligned}$$

THEOREM: elim-hyp-1a-3aa

$$\begin{aligned} & (\text{tcp}(x) \wedge \text{negativep}(x) \wedge (y \in \mathbf{N}) \wedge (y \neq 0) \wedge \text{imultiplep}(x, y)) \\ \rightarrow & \text{negativep}(\text{divide}(x, y)) \end{aligned}$$

THEOREM: elim-hyp-1a-3ab

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

THEOREM: transfer-add

$$(\text{tcp}(b) \wedge \text{tcp}(c)) \rightarrow ((\text{add}(a, b) = c) = (b = \text{add}(c, \text{tc-minus}(a))))$$

THEOREM: equal-multiplep-r-new-r

$$\begin{aligned} & (\text{tcp}(r) \wedge \text{tcp}(n) \wedge (i \in \mathbf{N})) \\ \rightarrow & (\text{imultiplep}(\text{new-r}(i, r, n), n) = \text{imultiplep}(r, n)) \end{aligned}$$

THEOREM: add-a-minus-a

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

THEOREM: tc-minus-tc-minus-a  
 $\text{tc-minus}(\text{tc-minus}(a)) = \text{tc-fix}(a)$

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

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

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

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

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

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

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

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

$$\begin{aligned}
& \wedge (n \in \mathbf{N}) \\
& \wedge (i \in \mathbf{N}) \\
& \wedge (n \neq 0) \\
& \wedge (i \neq 0) \\
& \wedge (r \not\prec (n * \text{twoto}(i))) \\
& \rightarrow (\text{new-r}(i, r, n) \not\prec (n * \text{twoto}(i - 1)))
\end{aligned}$$

THEOREM: elim-hyp-2a-4aa

$$\begin{aligned}
& ((r \in \mathbf{N}) \\
& \wedge \text{tcp}(n) \\
& \wedge \text{negativep}(n) \\
& \wedge \text{div-in-rangep}(r, n, i) \\
& \wedge \text{imultiplep}(\text{new-r}(i, r, n), n)) \\
& \rightarrow (\text{add}(\text{divide}(\text{new-r}(i, r, n), n), -1) \in \mathbf{N})
\end{aligned}$$

THEOREM: elim-hyp-2a-4ab

$$\begin{aligned}
& ((r \in \mathbf{N}) \wedge \text{tcp}(n) \wedge \text{negativep}(n) \wedge \text{div-in-rangep}(r, n, i)) \\
& \rightarrow (\text{divide}(\text{new-r}(i, r, n), n) \in \mathbf{N})
\end{aligned}$$

THEOREM: remainder-0-implies

$$\begin{aligned}
& ((a \in \mathbf{N}) \\
& \wedge (b \in \mathbf{N}) \\
& \wedge (c \in \mathbf{N}) \\
& \wedge (a \neq 0) \\
& \wedge ((c \bmod a) = 0) \\
& \wedge ((a * b) = c)) \\
& \rightarrow ((c \div a) = b)
\end{aligned}$$

THEOREM: elim-hyp-2a-4ba

$$\begin{aligned}
& (\text{negativep}(x) \\
& \wedge \text{tcp}(n) \\
& \wedge \text{negativep}(n) \\
& \wedge \text{imultiplep}(x, n) \\
& \wedge ((\text{negative-guts}(n) * \text{twoto}(i - 1)) \not\prec \text{negative-guts}(x))) \\
& \rightarrow \text{negativep}(\text{add}(\text{divide}(x, n), \text{add}(-1, \text{tc-minus}(\text{twoto}(i - 1)))))
\end{aligned}$$

THEOREM: equal-times-ab-c-equal-rem-ca-0

$$\begin{aligned}
& ((a \in \mathbf{N}) \wedge (b \in \mathbf{N}) \wedge (c \in \mathbf{N}) \wedge (a \neq 0) \wedge ((a * b) = c)) \\
& \rightarrow ((c \bmod a) = 0)
\end{aligned}$$

THEOREM: elim-hyp-2a-4bb

$$\begin{aligned}
& (\text{negativep}(x) \\
& \wedge \text{tcp}(n) \\
& \wedge \text{negativep}(n) \\
& \wedge (\neg \text{imultiplep}(x, n)) \\
& \wedge ((\text{negative-guts}(n) * \text{twoto}(i - 1)) \not\prec \text{negative-guts}(x))) \\
& \rightarrow \text{negativep}(\text{add}(\text{divide}(x, n), \text{tc-minus}(\text{twoto}(i - 1)))))
\end{aligned}$$



THEOREM: elim-hyp-2a-4c

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

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

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

THEOREM: elim-hyp-2b-2a

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

THEOREM: elim-hyp-2b-2b

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

THEOREM: elim-hyp-3aab-1c

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

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

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

THEOREM: elim-hyp-3ba-3c

$$\begin{aligned} & (\text{tcp}(r) \\ & \wedge \text{negativep}(r) \\ & \wedge (n \in \mathbf{N}) \\ & \wedge (n \neq 0)) \end{aligned}$$

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

THEOREM: elim-hyp-3c-3c

$$\begin{aligned}
& (\text{tcp}(r) \\
& \wedge \text{negativep}(r) \\
& \wedge (n \in \mathbf{N}) \\
& \wedge (n \neq 0) \\
& \wedge (i \in \mathbf{N}) \\
& \wedge ((n * \text{twoto}(i)) < \text{negative-guts}(r)) \\
& \wedge (i \neq 0)) \\
& \rightarrow ((n * \text{twoto}(i - 1)) < \text{negative-guts}(\text{new-r}(i, r, n)))
\end{aligned}$$

THEOREM: elim-hyp-4aa-2c

$$\begin{aligned}
& (\text{tcp}(r) \\
& \wedge \text{negativep}(r) \\
& \wedge \text{tcp}(n) \\
& \wedge \text{negativep}(n) \\
& \wedge \text{div-in-rangep}(r, n, i) \\
& \wedge (i \in \mathbf{N})) \\
& \rightarrow (\text{new-r}(i, r, n) < (\text{twoto}(i - 1) * \text{negative-guts}(n)))
\end{aligned}$$

THEOREM: div-not-in-range-sign-new-r-4

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

THEOREM: elim-hyp-4ba-4c

$$\begin{aligned}
& (\text{tcp}(r) \\
& \wedge \text{negativep}(r) \\
& \wedge \text{tcp}(n) \\
& \wedge \text{negativep}(n) \\
& \wedge (i \in \mathbf{N}) \\
& \wedge ((\text{twoto}(i) * \text{negative-guts}(n)) \not\leq \text{negative-guts}(r))) \\
& \rightarrow ((\text{twoto}(i - 1) * \text{negative-guts}(n)) \not\leq \text{negative-guts}(\text{new-r}(i, r, n)))
\end{aligned}$$

THEOREM: elim-hyp-4c-4c

$$\begin{aligned}
& (\text{tcp}(r) \\
& \wedge \text{negativep}(r) \\
& \wedge \text{tcp}(n)
\end{aligned}$$

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

THEOREM: `r-lessp-2n-quotient-1`

(( $n \neq 0$ )  $\wedge$  ( $r \not< n$ )  $\wedge$  ( $r < (2 * n)$ ))  $\rightarrow$  (( $r \div n = 1$ )

THEOREM: `r-lessp-equal-2n-quotient-2-aux`

$((n \in \mathbf{N})$   
 $\wedge$  ( $n \neq 0$ )  
 $\wedge$  ( $n < r$ )  
 $\wedge$  (( $2 * n \not< r$ )  
 $\wedge$  (( $r \bmod n = 0$ ))  
 $\rightarrow$  (( $2 * n = r$ )

THEOREM: `r-lessp-equal-2n-quotient-2`

$((n \in \mathbf{N})$   
 $\wedge$  ( $n \neq 0$ )  
 $\wedge$  ( $n < r$ )  
 $\wedge$  (( $2 * n \not< r$ )  
 $\wedge$  (( $r \bmod n = 0$ ))  
 $\rightarrow$  (( $r \div n = 2$ )

THEOREM: `not-lessp-x-n-equal-quotient-1`

$((x \in \mathbf{N})$   
 $\wedge$  ( $x \neq 0$ )  
 $\wedge$  ( $n \in \mathbf{N}$ )  
 $\wedge$  ( $n \neq 0$ )  
 $\wedge$  ( $n \not< x$ )  
 $\wedge$  (( $x \bmod n = 0$ ))  
 $\rightarrow$  (( $x \div n = 1$ )

THEOREM: `remainder-times-commuted`

(( $j * i \bmod i = 0$ )

THEOREM: `anrd-integer-ok-i-1`

$(\text{tcp}(r) \wedge \text{tcp}(n) \wedge (n \neq 0) \wedge (i = 1))$   
 $\rightarrow$  (`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**  $r < (n * \text{twoto}(i))$

```

        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 ∈ 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)) ≰ 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)

```

THEOREM: elim-hyp-2b-2c-corr

```

((r ∈ N)
 ∧ tcp (n)
 ∧ negativep (n)
 ∧ (r < (twoto (i) * negative-guts (n))))
→ (new-r (i, r, n) < (twoto (i - 1) * negative-guts (n)))

```

THEOREM: elim-hyp-2c-2c-corr

```

((r ∈ N)
 ∧ tcp (n)
 ∧ negativep (n)
 ∧ (i ∈ N)
 ∧ (i ≠ 0)
 ∧ (r ≰ (twoto (i) * negative-guts (n))))
→ (new-r (i, r, n) ≰ (twoto (i - 1) * negative-guts (n)))

```

THEOREM: lessp-twoto-sub1

```

((i ∈ N) ∧ ((n * twoto (i)) < x)) → ((n * twoto (i - 1)) < x)

```

THEOREM: elim-hyp-4ba-4aa-corr-aux

```

((r ∈ N)
 ∧ (n ∈ N)

```

$\wedge (n \neq 0)$   
 $\wedge (r \neq 0)$   
 $\wedge ((r \bmod n) = 0)$   
 $\wedge ((n * i) < r)$   
 $\wedge (i \neq 0)$   
 $\wedge (i \in \mathbf{N})$   
 $\rightarrow (((r \div n) - i) \neq 1)$

THEOREM: elim-hyp-4ba-4aa-corr

$(\text{tcp}(r))$   
 $\wedge \text{negativep}(r)$   
 $\wedge \text{tcp}(n)$   
 $\wedge \text{negativep}(n)$   
 $\wedge \text{imultiplep}(r, n)$   
 $\wedge (\neg \text{div-in-rangep}(r, n, i))$   
 $\rightarrow (\text{add}(\text{divide}(\text{new-r}(i, r, n), n), -1) \in \mathbf{N})$

THEOREM: elim-hyp-4bb-4ab-corr

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

THEOREM: anrd-integer-ok

$(\text{tcp}(r) \wedge \text{tcp}(n) \wedge (n \neq 0) \wedge (i \in \mathbf{N}))$   
 $\rightarrow (\text{tc-to-integer}(\text{anrd}(i, r, n))$   
 $\quad = \text{if } i = 0 \text{ then } 0$   
 $\quad \quad \text{elseif } r \in \mathbf{N}$   
 $\quad \quad \text{then if } n \in \mathbf{N}$   
 $\quad \quad \quad \text{then if div-in-rangep}(r, n, i) \text{ then divide}(r, n)$   
 $\quad \quad \quad \quad \text{elseif } r < (n * \text{twoto}(i))$   
 $\quad \quad \quad \quad \quad \text{then add}(\text{divide}(r, n), \text{tc-minus}(\text{twoto}(i)))$   
 $\quad \quad \quad \quad \quad \text{else } -1 \text{ endif}$   
 $\quad \quad \quad \text{elseif div-in-rangep}(r, n, i) \text{ then add}(\text{divide}(r, n), -1)$   
 $\quad \quad \quad \text{elseif } r < (\text{negative-guts}(n) * \text{twoto}(i))$   
 $\quad \quad \quad \quad \text{then add}(\text{divide}(r, n), \text{add}(\text{twoto}(i), -1))$   
 $\quad \quad \quad \quad \text{else } 0 \text{ endif}$   
 $\quad \quad \text{elseif } n \in \mathbf{N}$   
 $\quad \quad \text{then if div-in-rangep}(r, n, i)$   
 $\quad \quad \quad \text{then if imultiplep}(r, n) \text{ then divide}(r, n)$   
 $\quad \quad \quad \quad \text{else add}(\text{divide}(r, n), -1) \text{ endif}$   
 $\quad \quad \quad \text{elseif } (n * \text{twoto}(i)) \neq \text{negative-guts}(r)$   
 $\quad \quad \quad \quad \text{then if imultiplep}(r, n) \text{ then add}(\text{divide}(r, n), \text{twoto}(i))$

```

        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)) <math>\not\leq</math> 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 ∈  $\mathbf{N}$ ) = (r ∈  $\mathbf{N}$ ) 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 ≈ 0 then BV-NIL
    else bitvec ((noem ∈  $\mathbf{N}$ ) = (long-new-r (r, noem) ∈  $\mathbf{N}$ ),
        long-nrd (i - 1, mult (2, long-new-r (r, noem)), noem)) endif

```

DEFINITION:

```

alt-long-new-r (r, noem)
= if (r ∈  $\mathbf{N}$ ) = (noem ∈  $\mathbf{N}$ ) then add (mult (2, r), tc-minus (noem))
    else add (mult (2, r), noem) endif

```

DEFINITION:

```

alt-long-nrd (i, r, noem)
= if i ≈ 0 then BV-NIL
    else bitvec ((noem ∈  $\mathbf{N}$ ) = (alt-long-new-r (r, noem) ∈  $\mathbf{N}$ ),
        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 ≈ 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 ( $r$ )  
 $\wedge$  bitvecp ( $noem$ )  
 $\wedge$  (bv-size ( $r$ ) = bv-size ( $noem$ ))  
 $\wedge$  evenp (bv-size ( $r$ ))  
 $\wedge$  ( $noem \neq \text{BV-NIL}$ )  
 $\rightarrow$  (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 ( $r$ ,  $\mathbf{f}$ ),  $n$ , 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 ( $r$ ,  $\mathbf{f}$ ),  $n$ , 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  
 ((( $a \in \mathbf{N}$ ) = ( $b \in \mathbf{N}$ ))  
 $\wedge$  tcp ( $a$ )  
 $\wedge$  tc-in-rangep ( $a$ ,  $n$ )  
 $\wedge$  tcp ( $b$ )  
 $\wedge$  tc-in-rangep ( $b$ ,  $n$ )  
 $\rightarrow$  tc-in-rangep (add ( $a$ , tc-minus ( $b$ )),  $n$ ))

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



THEOREM: size-mcalu

(bitvecp ( $a$ )  
  $\wedge$  bitvecp ( $b$ )  
  $\wedge$  boolp ( $byp$ )  
  $\wedge$  boolp ( $ccin$ )  
  $\wedge$  controlep ( $cr$ )  
  $\wedge$  (bv-size ( $a$ ) = bv-size ( $b$ ))  
  $\wedge$  evenp (bv-size ( $a$ ))  
  $\wedge$  ( $b \neq \text{BV-NIL}$ )  
  $\wedge$  controlep ( $cp$ )  
  $\wedge$  controlep ( $ck$ )  
  $\rightarrow$  (bv-size (csobv-bitvec (mcalu (bv-size ( $a$ ),  $byp$ ,  $a$ ,  $b$ ,  $cp$ ,  $ck$ ,  $cr$ ,  $ccin$ )))  
 = bv-size ( $a$ ))

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

(tc-to-integer ( $tel$ )  $\in \mathbf{N}$ )  
  $\rightarrow$  (( $2 * \text{tc-to-integer} (tel)$ ) < twoto (bv-size ( $tel$ )))

THEOREM: special-francky-third-1

((abs (tc-to-integer ( $noem$ ))  $\not\leq$  abs (tc-to-integer ( $tel$ )))  
  $\wedge$  ( $\neg$  tc-in-rangep (mult (2, tc-to-integer ( $tel$ )), bv-size ( $tel$ )))  
  $\wedge$  (tc-to-integer ( $noem$ )  $\in \mathbf{N}$ )  
  $\wedge$  (tc-to-integer ( $tel$ )  $\in \mathbf{N}$ )  
  $\rightarrow$  ( $\neg$  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-1bis

((abs (tc-to-integer ( $noem$ ))  $\not\leq$  abs (tc-to-integer ( $tel$ )))  
  $\wedge$  bitvecp ( $tel$ )  
  $\wedge$  bitvecp ( $noem$ )  
  $\wedge$  ( $\neg$  tc-in-rangep (mult (2, tc-to-integer ( $tel$ )), bv-size ( $tel$ )))  
  $\wedge$  (tc-to-integer ( $noem$ )  $\in \mathbf{N}$ )  
  $\wedge$  (tc-to-integer ( $tel$ )  $\in \mathbf{N}$ )  
  $\rightarrow$  negativep (add (add (mult (2, tc-to-integer ( $tel$ )),  
 tc-minus (tc-to-integer ( $noem$ ))),  
 - twoto (bv-size ( $tel$ ))))

THEOREM: special-francky-third-2

((abs (tc-to-integer ( $noem$ ))  $\not\leq$  abs (tc-to-integer ( $tel$ )))  
  $\wedge$  ( $\neg$  tc-in-rangep (mult (2, tc-to-integer ( $tel$ )), bv-size ( $tel$ )))  
  $\wedge$  negativep (tc-to-integer ( $noem$ ))  
  $\wedge$  (tc-to-integer ( $tel$ )  $\in \mathbf{N}$ )  
  $\rightarrow$  ( $\neg$  tc-in-rangep (add (add (mult (2, tc-to-integer ( $tel$ )),

$$\begin{aligned} & \text{tc-to-integer } (noem), \\ & - \text{twoto } (\text{bv-size } (tel)), \\ & \text{bv-size } (tel) \end{aligned}$$

THEOREM: special-francky-third-2-bis

$$\begin{aligned} & ((\text{abs } (\text{tc-to-integer } (noem)) \not\leq \text{abs } (\text{tc-to-integer } (tel))) \\ & \wedge \text{bitvecp } (tel) \\ & \wedge \text{bitvecp } (noem) \\ & \wedge (\neg \text{tc-in-rangep } (\text{mult } (2, \text{tc-to-integer } (tel)), \text{bv-size } (tel))) \\ & \wedge \text{negativep } (\text{tc-to-integer } (noem)) \\ & \wedge (\text{tc-to-integer } (tel) \in \mathbf{N}) \\ & \rightarrow \text{negativep } (\text{add } (\text{add } (\text{mult } (2, \text{tc-to-integer } (tel)), \text{tc-to-integer } (noem)), \\ & \quad - \text{twoto } (\text{bv-size } (tel)))) \end{aligned}$$

THEOREM: special-francky-third-3

$$\begin{aligned} & ((\text{abs } (\text{tc-to-integer } (noem)) \not\leq \text{abs } (\text{tc-to-integer } (tel))) \\ & \wedge (\neg \text{tc-in-rangep } (\text{mult } (2, \text{tc-to-integer } (tel)), \text{bv-size } (tel))) \\ & \wedge \text{negativep } (\text{tc-to-integer } (noem)) \\ & \wedge \text{negativep } (\text{tc-to-integer } (tel))) \\ & \rightarrow (\neg \text{tc-in-rangep } (\text{add } (\text{add } (\text{mult } (2, \text{tc-to-integer } (tel)), \\ & \quad \text{tc-minus } (\text{tc-to-integer } (noem))), \\ & \quad \text{twoto } (\text{bv-size } (tel))), \\ & \quad \text{bv-size } (tel))) \end{aligned}$$

THEOREM: special-francky-third-3-bis

$$\begin{aligned} & ((\text{abs } (\text{tc-to-integer } (noem)) \not\leq \text{abs } (\text{tc-to-integer } (tel))) \\ & \wedge \text{bitvecp } (tel) \\ & \wedge \text{bitvecp } (noem) \\ & \wedge (\neg \text{tc-in-rangep } (\text{mult } (2, \text{tc-to-integer } (tel)), \text{bv-size } (tel))) \\ & \wedge \text{negativep } (\text{tc-to-integer } (noem)) \\ & \wedge \text{negativep } (\text{tc-to-integer } (tel))) \\ & \rightarrow (\text{add } (\text{add } (\text{mult } (2, \text{tc-to-integer } (tel)), \text{tc-minus } (\text{tc-to-integer } (noem))), \\ & \quad \text{twoto } (\text{bv-size } (tel))) \in \mathbf{N}) \end{aligned}$$

THEOREM: special-francky-third-4

$$\begin{aligned} & ((\text{abs } (\text{tc-to-integer } (noem)) \not\leq \text{abs } (\text{tc-to-integer } (tel))) \\ & \wedge (\neg \text{tc-in-rangep } (\text{mult } (2, \text{tc-to-integer } (tel)), \text{bv-size } (tel))) \\ & \wedge (\text{tc-to-integer } (noem) \in \mathbf{N}) \\ & \wedge \text{negativep } (\text{tc-to-integer } (tel))) \\ & \rightarrow (\neg \text{tc-in-rangep } (\text{add } (\text{add } (\text{mult } (2, \text{tc-to-integer } (tel)), \\ & \quad \text{tc-to-integer } (noem)), \\ & \quad \text{twoto } (\text{bv-size } (tel))), \\ & \quad \text{bv-size } (tel))) \end{aligned}$$

THEOREM: special-francky-third-4-bis

$$\begin{aligned}
& ((\text{abs}(\text{tc-to-integer}(\text{noem})) \not\leq \text{abs}(\text{tc-to-integer}(\text{tel}))) \\
& \wedge \text{bitvecp}(\text{tel}) \\
& \wedge \text{bitvecp}(\text{noem}) \\
& \wedge (\neg \text{tc-in-range}(\text{mult}(2, \text{tc-to-integer}(\text{tel})), \text{bv-size}(\text{tel}))) \\
& \wedge (\text{tc-to-integer}(\text{noem}) \in \mathbf{N}) \\
& \wedge \text{negativep}(\text{tc-to-integer}(\text{tel})) \\
& \rightarrow (\text{add}(\text{add}(\text{mult}(2, \text{tc-to-integer}(\text{tel})), \text{tc-to-integer}(\text{noem})), \\
& \quad \text{twoto}(\text{bv-size}(\text{tel}))) \in \mathbf{N})
\end{aligned}$$

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

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

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

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

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

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

DEFINITION:  
 $\text{new-z}(z, \text{noem}, \text{prev})$   
 $=$  **if**  $\text{prev} = \text{bv-bit}(\text{noem})$   
**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 ∈ 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)

```

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

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

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

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

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

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

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

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

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

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

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

THEOREM: not-zero-bv  
 $\text{bv-not}(\text{zero-bitvec}(nb)) = \text{one-bitvec}(nb)$

THEOREM: haiblnr-1  
 $(\text{bitvecp}(ps)$   
 $\wedge \text{bitvecp}(z)$   
 $\wedge \text{bitvecp}(n)$   
 $\wedge (\text{bv-size}(n) = \text{bv-size}(z))$   
 $\wedge (\text{bv-size}(ps) = \text{bv-size}(z))$   
 $\wedge (n \neq \text{BV-NIL})$   
 $\wedge \text{evenp}(\text{bv-size}(z)))$   
 $\rightarrow (\text{bv-bit}(\text{new-z}(z, n, prev))$   
 $\quad = \text{bv-bit}(\text{bv-long-new-r}(\text{bv-append}(z, ps),$   
 $\quad \text{bv-append}(n, \text{zero-bitvec}(\text{bv-size}(z))),$   
 $\quad prev)))$

THEOREM: append-distributes  
 $(\text{bitvecp}(a) \wedge \text{bitvecp}(b) \wedge \text{bitvecp}(c))$   
 $\rightarrow (\text{bv-append}(\text{bv-append}(a, b), c) = \text{bv-append}(a, \text{bv-append}(b, c)))$

THEOREM: haiblnr-2  
 $(\text{bitvecp}(z)$   
 $\wedge \text{bitvecp}(n)$   
 $\wedge \text{bitvecp}(ps)$   
 $\wedge \text{boolp}(prev)$   
 $\wedge (\text{bv-size}(z) = \text{bv-size}(n))$   
 $\wedge (\text{bv-size}(z) = \text{bv-size}(ps))$   
 $\wedge (n \neq \text{BV-NIL})$   
 $\wedge \text{evenp}(\text{bv-size}(z)))$   
 $\rightarrow (\text{bv-append}(\text{up-shift-1}(\text{new-z}(z, n, prev), \text{bv-bit}(ps)), \text{up-shift-1}(ps, \mathbf{f}))$   
 $\quad = \text{up-shift-1}(\text{bv-long-new-r}(\text{bv-append}(z, ps),$   
 $\quad \text{bv-append}(n, \text{zero-bitvec}(\text{bv-size}(z))),$   
 $\quad prev),$   
 $\quad \mathbf{f}))$

THEOREM: hard-anrd-it-bv-long-nrd-relate  
 $(\text{bitvecp}(ps)$   
 $\wedge \text{bitvecp}(z)$

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

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

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

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

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

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

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

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

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

DEFINITION:  
 $\text{simple-induct}(x)$   
 $= \text{if } x \simeq 0 \text{ then } t$   
 $\quad \text{else simple-induct}(x - 1) \text{ endif}$

THEOREM: up-shift-1-append  
 $(\text{bitvecp}(x) \wedge \text{bitvecp}(y) \wedge (y \neq \text{BV-NIL}) \wedge \text{boolp}(sin))$   
 $\rightarrow (\text{up-shift-1}(\text{bv-append}(x, y), sin))$   
 $= \text{bv-append}(\text{up-shift-1}(x, \text{bv-bit}(y)), \text{up-shift-1}(y, sin))$

THEOREM: append-butlast-last  
 $(\text{bitvecp}(y) \wedge (y \neq \text{BV-NIL}))$   
 $\rightarrow (\text{bv-append}(\text{butlast}(y), \text{bitvec}(\text{last}(y), \text{BV-NIL})) = y)$

THEOREM: us1-dos-1  
 $(\text{bitvecp}(y) \wedge \text{boolp}(dsin))$   
 $\rightarrow (\text{up-shift-1}(\text{do-shift-1}(y, dsin), \text{last}(y)) = y)$

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

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

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

THEOREM: first-size-x  
 $\text{bitvecp}(x) \rightarrow (\text{first}(\text{bv-size}(x), x) = x)$

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

THEOREM: butlast-bitvec  
 $(\text{bitvecp}(rest) \wedge (rest \neq \text{BV-NIL}))$   
 $\rightarrow (\text{butlast}(\text{bitvec}(bit, rest)) = \text{bitvec}(bit, \text{butlast}(rest)))$

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

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



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

$$\begin{aligned} &(\text{up-shift-n}(n, x, sin) = \text{BV-NIL}) \\ &= ((\neg \text{boolp}(sin)) \vee (\neg \text{bitvecp}(x)) \vee (n \notin \mathbf{N}) \vee (x = \text{BV-NIL})) \end{aligned}$$

THEOREM: last-first

$$\begin{aligned} &(\text{boolp}(sin) \wedge (m \in \mathbf{N}) \wedge (m \neq 0) \wedge \text{bitvecp}(x) \wedge (\text{bv-size}(x) \not\prec m)) \\ &\rightarrow (\text{last}(\text{first}(m, x)) = \text{bv-bit}(\text{up-shift-n}(m - 1, x, sin))) \end{aligned}$$

THEOREM: last-id-bitvec

$$(\text{boolp}(bit) \wedge (n \neq 0)) \rightarrow (\text{last}(\text{id-bitvec}(n, bit)) = bit)$$

THEOREM: bit-us-last-dos

$$\begin{aligned} &(\text{bitvecp}(x) \wedge (m \in \mathbf{N}) \wedge (\text{bv-size}(x) \not\prec m)) \\ &\rightarrow (\text{last}(\text{do-shift-n}(\text{bv-size}(x) - m, x, \text{bv-bit}(x))) \\ &= \text{bv-bit}(\text{up-shift-n}(m - 1, x, \mathbf{f}))) \end{aligned}$$

THEOREM: append-of-up-shift

$$\begin{aligned} &(\text{bitvecp}(x) \wedge (l \in \mathbf{N}) \wedge (\text{bv-size}(x) = nb) \wedge (nb \not\prec l)) \\ &\rightarrow (\text{bv-append}(\text{do-shift-n}(nb - l, x, \text{bv-bit}(x)), \text{up-shift-n}(l, x, \mathbf{f})) \\ &= \text{up-shift-n}(l, \text{bv-append}(\text{id-bitvec}(nb, \text{bv-bit}(x)), x), \mathbf{f})) \end{aligned}$$

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

$$\text{tcp}(x) \rightarrow (\text{mult}(\mathbf{1}, x) = x)$$

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  
 $\text{tc-to-integer}(\text{bv-append}(x, \text{bitvec}(\mathbf{f}, \text{BV-NIL}))) = \text{mult}(2, \text{tc-to-integer}(x))$

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

THEOREM: times-conserves-lessp  
 $((a \in \mathbf{N}) \wedge (b \in \mathbf{N}) \wedge (c \in \mathbf{N}) \wedge (b \neq 0) \wedge (c \not\leq a))$   
 $\rightarrow ((b * c) \not\leq a)$

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

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

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

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

THEOREM: times-conserves-lessp-real  
 $((a \in \mathbf{N}) \wedge (b \in \mathbf{N}) \wedge (c \in \mathbf{N}) \wedge (b \neq 0) \wedge (a < c))$   
 $\rightarrow (a < (b * c))$

THEOREM: times-conserves-lessp-dual-real  
 $((a \in \mathbf{N}) \wedge (b \in \mathbf{N}) \wedge (c \in \mathbf{N}) \wedge (b \neq 0) \wedge (a < c))$   
 $\rightarrow (a < (c * b))$

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

THEOREM: div-in-rangep-tc-to-ints  
 (bitvecp (*tel*)  
 ∧ bitvecp (*noem*)  
 ∧ (bv-size (*tel*) = bv-size (*noem*))  
 ∧ (tc-to-integer (*noem*) ≠ 0))  
 → div-in-rangep (tc-to-integer (*tel*), tc-to-integer (*noem*), bv-size (*tel*))

THEOREM: hard-anrd-init-divides  
 (bitvecp (*tel*)  
 ∧ bitvecp (*noem*)  
 ∧ (bv-size (*tel*) = bv-size (*noem*))  
 ∧ (bv-size (*tel*) = *nb*)  
 ∧ evenp (bv-size (*tel*))  
 ∧ (*noem* ≠ BV-NIL)  
 ∧ (tc-to-integer (*noem*) ≠ 0))  
 → (tc-to-integer (hard-anrd-init (*nb*, 1, *tel*, *noem*))  
 = **if** tc-to-integer (*tel*) ∈ **N**  
   **then if** tc-to-integer (*noem*) ∈ **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*) ∈ **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

- abs, 37, 57–59, 66
- abs-upper-bound-*tc-to-int*, 66
- add, 9–11, 13, 30, 31, 33, 34, 37, 38, 40, 41, 43–49, 52–54, 56–59, 63, 67
- add-0, 31
- add-1-1, 31
- add-a-minus-a, 46
- addbyp-fcout, 13, 15, 16
- adder-with-one-bv, 61
- adder-with-zero-bv, 61
- alt-bv-long-new-r, 55, 56, 59
- alt-bv-long-new-r-alt-long-new-r-relate, 59
- alt-bv-long-new-r-bv-long-new-r-relate, 56
- alt-bv-long-nrd, 56, 59
- alt-bv-long-nrd-alt-long-nrd-relate, 59
- alt-bv-long-nrd-bv-long-nrd-relate, 56
- alt-long-new-r, 54, 59
- alt-long-new-r-long-new-r-relate, 54
- alt-long-nrd, 54, 59
- alt-long-nrd-long-nrd-relate, 54
- alucev-fc, 13–17, 19, 20, 22
- alucod-fc, 13–17, 19, 20, 22
- alugen-alugenod-relate, 23
- alugen-o, 22–25
- alugenod-o, 23, 25
- alugenodo-func, 23
- anrd, 40, 51, 53, 63, 66
- anrd-i-0, 40
- anrd-integer-ok, 53
- anrd-integer-ok-i-1, 51
- app-c4-carry1, 19
- app-c4-carry2, 19
- append-butlast-last, 64
- append-distributes, 62
- append-not-nil, 35
- append-not-nil-2, 59
- append-not-nil-3, 64
- append-of-up-shift, 65
- associativity-of-add, 9
- associativity-of-mult, 37
- associativity-of-plus, 2
- associativity-of-times, 2
- bit-of-bv-not, 31
- bit-on-implies-non-0, 11
- bit-us-last-dos, 65
- bitvec, 6–12, 14, 15, 17–19, 22, 24, 25, 35, 39, 40, 54–56, 59–61, 64–66
- bitvec-carry-ovf, 8, 11, 12, 14, 15, 17–19, 21, 22, 26, 27
- bitvecp, 6–13, 16, 17, 19–21, 23–40, 56–67
- boolp, 6, 8, 10–17, 19–21, 23, 25–30, 32–36, 57, 60, 62–65
- boolp-alucev-fc, 13
- boolp-alucod-fc, 13
- boolp-bv-bit, 6
- boolp-bvco-carry, 8
- boolp-bvco-ovf, 8
- boolp-not, 30
- butlast, 35, 39, 64
- butlast-bitvec, 64
- butlast-bv-append, 39
- butlast-first, 64
- bv-0, 6, 24, 25
- bv-1, 6, 24, 25
- bv-2, 6, 24, 25
- bv-3, 6, 23–25
- bv-adder, 11–13, 28–30, 34, 61, 62
- bv-adder-non-nil, 12
- bv-adder-non-nil2, 34
- bv-adder-plusses, 12
- bv-append, 8, 18, 19, 21, 35–37, 39, 40, 59, 61–66
- bv-append-bitvec, 61
- bv-append-not-not, 36

- bv-append-of-bv-nil, 39
- bv-bit, 6–12, 17–22, 24, 25, 27, 31, 35, 37, 39, 55, 56, 59–62, 64, 65
- bv-bit-of-bv-append, 39
- bv-exor, 7, 8, 24, 25, 28, 29, 31
- bv-exor-nil-a, 7
- bv-exor-nil-b, 7
- bv-invert-even, 7, 8, 25, 29
- bv-long-new-r, 55, 56, 62
- bv-long-nrd, 55, 56, 63
- bv-nil, 6–15, 17, 20, 21, 23–25, 28–36, 38–40, 54–57, 59, 60, 62–67
- bv-not, 7, 8, 24, 25, 28–32, 36, 62
- bv-not-bv-exor-right, 31
- bv-not-nil, 31
- bv-size, 6–13, 17–38, 40, 43, 55–67
- bv-size-alt-long-new-r, 56
- bv-to-nat, 9–12, 30, 31, 37, 39, 43
- bv-to-nat-bv-append, 37
- bv-to-nat-of-bv-not, 30
- bv-to-nat-of-vec, 37
- bv-to-nat-to-integer, 11
- bv-to-nat-to-integer-lemma2, 10
- bv-vec, 6–9, 11, 12, 17–22, 24, 25, 28, 35–37, 64, 65
- bvco-bitvec, 11–13, 17–19, 21, 22, 26–30, 34, 61, 62
- bvco-bitvec-bit-adders-double-si
  - ze, 61
  - ze-t, 62
- bvco-carry, 8, 11, 12, 17–22, 26–28, 61
- bvco-carry-bit-adders-double-si
  - ze, 61
  - ze-t, 61
- bvco-nil, 12, 14, 15, 17, 19, 22, 27
- bvco-ovf, 8, 26–29
  
- c4, 13, 16, 18–20
- c4-c4byp-help, 15
- c4-c4byp-relate, 16
- c4byp, 14, 16, 21
  
- carry, 6, 10–13, 30, 31, 33, 34
- carry-adder-with-one-bv, 61
- carry-adder-with-zero-bv, 61
- carry-col, 16, 17, 19–22, 26–29
- carry-col2, 17–19, 21, 22
- carry-not, 31
- carry-sign-ovf-bitvec, 8, 28
- carry2-carry-relate, 21
- cbyp-carry-relate, 22
- cbyp-carry2-relate, 22
- cbyp-col, 21, 22, 26
- commutativity-of-add, 9
- commutativity-of-mult, 37
- commutativity-of-plus, 2
- commutativity-of-times, 2
- commutativity2-of-add, 9
- commutativity2-of-mult, 37
- commutativity2-of-plus, 2
- commutativity2-of-times, 2
- controlep, 6, 23–25, 27, 57
- cscbi1, 23–25, 27, 28
- cscbo1, 23, 27
- csexor, 27, 28
- csobv-bitvec, 29, 30, 33, 34, 55–57, 60
- csobv-nil, 28
  
- difference-1, 44
- difference-2, 3
- difference-add1, 4
- difference-crock1, 3
- difference-difference, 3
- difference-elim, 3
- difference-plus, 3
- difference-plus-cancellation, 3
- difference-x-x, 3
- distributivity-of-times-over-plus, 2
- div-in-range-sign-new-r-1, 46
- div-in-range-sign-new-r-2, 46
- div-in-range-sign-new-r-3, 46
- div-in-range-sign-new-r-4, 46
- div-in-rangep, 43, 44, 46–54, 67
- div-in-rangep-new-r-sign-1, 43

- div-in-rangep-new-r-sign-2, 43
- div-in-rangep-new-r-sign-3, 44
- div-in-rangep-new-r-sign-4, 44
- div-in-rangep-tc-to-ints, 67
- div-not-in-range-sign-new-r-1, 47
- div-not-in-range-sign-new-r-2, 49
- div-not-in-range-sign-new-r-3, 49
- div-not-in-range-sign-new-r-4, 50
- divide, 38, 45–49, 51–54, 67
- divide-add-mult, 46
- do-shift-0, 36
- do-shift-1, 35, 64
- do-shift-n, 35, 36, 61, 64, 65
- do-shift-n-first-append-relate, 64
- do-shift-strict, 36
  
- elim-hyp-1a-3aa, 46
- elim-hyp-1a-3ab, 46
- elim-hyp-1a-3ba, 47
- elim-hyp-1a-3bb, 47
- elim-hyp-1a-3c, 47
- elim-hyp-1b-1a, 47
- elim-hyp-1b-1b, 47
- elim-hyp-1b-1c, 47
- elim-hyp-1c, 47
- elim-hyp-2a-4aa, 48
- elim-hyp-2a-4ab, 48
- elim-hyp-2a-4ba, 48
- elim-hyp-2a-4bb, 48
- elim-hyp-2a-4c, 49
- elim-hyp-2b-2a, 49
- elim-hyp-2b-2b, 49
- elim-hyp-2b-2c-corr, 52
- elim-hyp-2c-2c-corr, 52
- elim-hyp-3aab-1c, 49
- elim-hyp-3ba-3c, 49
- elim-hyp-3c-3c, 50
- elim-hyp-4aa-2c, 50
- elim-hyp-4ba-4aa-corr, 53
- elim-hyp-4ba-4aa-corr-aux, 52
- elim-hyp-4ba-4c, 50
- elim-hyp-4bb-4ab-corr, 53
- elim-hyp-4c-4c, 50
- equal-bit-implies-equal-numberp, 56
  
- equal-bools, 2
- equal-diff-twoto, 39
- equal-difference-0, 31
- equal-multiplep-r-new-r, 46
- equal-numberp-implies-tc-in-range-sub, 56
- equal-times-0, 2
- equal-times-ab-c-equal-rem-ca-0, 48
- equal-times-times, 42
- equal-times-times-commuted, 45
- evenp, 5, 7, 17–22, 25–30, 32–34, 36, 39, 56, 57, 59, 60, 62, 63, 66, 67
- evenp-add1, 5
- evenp-add1-commuted, 39
- evenp-plus, 36
- evenp-plus-extended, 36
- evenp-times-even, 39
- evenp-twoto, 36
- exor, 6, 7, 11, 23, 27
  
- first, 64, 65
- first-size-x, 64
- first-strict, 65
- full-down-shift, 65
  
- hack-around-next-lemma, 42
- hack-implies-sprop, 61
- hack2, 36
- hackxaux, 38
- haiblnr-1, 62
- haiblnr-2, 62
- hard-anrd-anrd-relate-1, 66
- hard-anrd-init, 60, 66, 67
- hard-anrd-init-divides, 67
- hard-anrd-it, 60, 61, 63
- hard-anrd-it-bv-long-nrd-relate, 62
  
- id-bitvec, 64, 65
- id-bitvec-strict, 65
- imultiplep, 41–43, 46–48, 52–54, 67
- imultiplep-add, 41
- imultiplep-new-r, 42
- induct-carry-col2, 20

- induct-fn, 54
- induct-quot-quot, 40
- induct-vec-vec-evenp-f, 28
- integer-in-range-of-tc-to-integer, 10
- integer-interpretation-of-bv-adder-output, 13
  - der-output-lemma1, 12
- integer-interpretation-of-up-shift-1, 38
- integer-to-nat, 10, 11, 38
- invert-lsb, 39
  
- kill-col, 24–27
- kill-col-12, 25
  
- last, 34, 35, 39, 40, 64, 65
- last-first, 65
- last-id-bitvec, 65
- last-of-bv-append, 40
- lemma1, 25
- lessp-abs-max-upshift, 66
- lessp-boolp, 6
- lessp-difference, 4
- lessp-not-lessp-times, 63
- lessp-plus-1, 38
- lessp-plus-1-commuted, 38
- lessp-plus-2, 39
- lessp-remainder, 4
- lessp-times, 3
- lessp-twoto-sub1, 52
- long-new-r, 54, 63
- long-new-r-mult-2, 63
- long-new-r-new-r-relate, 63
- long-nrd, 54, 63
- long-nrd-anrd-relate, 63
- long-nrd-mult-2, 63
- lower-bound-on-negative-bv-to-nat, 10
- lower-upper-determines, 42
- lsb-implies-odd, 39
  
- may-be-baby, 44
- may-be-baby-2, 44
- may-be-baby-3, 44
- may-be-baby-4, 45
- may-be-baby-5, 45
- may-be-baby-6, 45
- mcalu, 27, 29, 30, 32–34, 55–57, 60
- mcalu-12-5-10-mcalu-12-10-10-relate, 32
- mcalu-imp-bv-adder-relate, 29
- mult, 36–38, 40, 41, 43, 44, 46, 54, 57–59, 63, 65, 66
- mult-1, 65
- mult-add-distributes, 63
- multiplep, 39, 41, 44
- multiplep-diff, 41
- multiplep-tc-minus, 42
  
- nat-interpretation-of-bv-adder-output, 12
- nat-to-bv, 9, 24, 25, 29, 30, 32–34, 55, 56, 60
- nat-to-bv-of-trunc, 10
- nat-to-integer, 10–12, 38
- neg-guts-tc-to-integer-0, 67
- negative-guts-tc-minus, 39
- negativep-mult, 59
- new-q, 59, 60
- new-r, 40, 42, 43, 46–53, 63
- new-z, 59, 60, 62
- not-bit-implies-in-range-times-2, 37
- not-equal-numberp-implies-tc-in-range-add, 56
- not-equal-times-0, 42
- not-even-add1-commuted, 39
- not-evenp-add1, 5
- not-imultiplep-add, 43
- not-imultiplep-new-r, 43
- not-lessp-not-lessp-times, 63
- not-lessp-x-n-equal-quotient-1, 51
- not-zero-bv, 62
- numberp-mult, 63
  
- one-bitvec, 7, 8, 61, 62
- pathological-difference, 3

- plus-1, 1
- plus-add1, 1
- plus-cancellation, 2
- plus-equal-0, 2
- plus-right-id, 1
- plus-to-add, 10
- prop-col, 23, 24, 26, 27
- prop-col-10, 24
- prop-col-12, 24
- prop-col-5, 24
- prop-kill-relate, 24
  
- q-d-t-lemma1, 41
- q-d-t-lemma2, 41
- q-d-t-lemma3, 41
- quotient-diff-times, 40
- quotient-diff-times-commuted, 41
- quotient-diff-times-commuted-dual, 44
- quotient-diff-times-dual, 44
- quotient-lessp, 40
- quotient-plus-times, 4
- quotient-plus-times-commuted, 4
- quotient-times, 5
- quotient-times-commuted, 41
- quotient-times-lessp, 41
- quotient-times-lessp-refrased, 44
  
- r-lessp-2n-quotient-1, 51
- r-lessp-equal-2n-quotient-2, 51
- r-lessp-equal-2n-quotient-2-aux, 51
- real-hack-1, 39
- real-hack-6, 39
- rec-mcalu-imp, 26-29
- rec-mcalu-imp-bv-adder-relate, 29
- remainder-0-implies, 48
- remainder-by-nonnumber, 4
- remainder-crock3, 5
- remainder-crock4, 5
- remainder-diff, 37
- remainder-diff-times, 37
- remainder-diff-times-commuted, 42
- remainder-diff-times-commuted-2, 43
- remainder-natural-interpretatio
  - n-of-up-shift-1, 37
- remainder-of-0, 5
- remainder-of-1, 5
- remainder-plus, 4
- remainder-plus-times, 4
- remainder-plus-times-commuted, 4
- remainder-quotient, 4
- remainder-quotient-elim, 4
- remainder-remainder, 37
- remainder-times, 4
- remainder-times-commuted, 51
- remainder-x-x, 4
- res-col, 25, 26
- res-col-10, 25
  
- simple-induct, 63
- size-0, 6
- size-anrd-bv, 40
- size-bv-adder, 12
- size-bv-append, 35
- size-bvexor, 8
- size-bvinv, 8
- size-bvnot, 8
- size-carry-col, 17
- size-do-shift-n, 36
- size-id-bitvec, 64
- size-kill-col, 24
- size-mcalu, 57
- size-new-z, 60
- size-onebv, 8
- size-prop-col, 24
- size-res-col, 25
- size-up-shift-1, 36
- size-up-shift-n, 36
- size-zerobv, 7
- special-francky-first-leq, 59
- special-francky-third-1, 57
- special-francky-third-1bis, 57
- special-francky-third-2, 57
- special-francky-third-2-bis, 58
- special-francky-third-3, 58
- special-francky-third-3-bis, 58
- special-francky-third-4, 58
- special-francky-third-4-bis, 59



- sub1-difference, 44
- tc-fix, 31, 47
- tc-in-range-mult-2-tc-to-int-1, 57
- tc-in-rangep, 9–11, 13, 30, 33, 34, 38, 56–59
- tc-interpretation-of-mcalu-imp-
  - add, 32
  - sub, 34
  - xsub, 33
- tc-interpretation-of-mcalu-outp
  - ut, 30
- tc-minus, 30–34, 36, 38–44, 46–48, 52–54, 56–58
- tc-minus-mult, 41
- tc-minus-tc-minus-a, 47
- tc-minus-tc-to-integer, 31
- tc-to-int-append-simple-zero-bitvec, 66
- tc-to-int-append-zero-bv, 66
- tc-to-int-sign-extension, 65
- tc-to-int-simple-sign-extension, 65
- tc-to-integer, 9–13, 30, 31, 33, 34, 38, 40, 43, 51, 53, 56–59, 65–67
- tc-to-integer-0, 31
- tc-to-integer-bv-not, 31
- tc-to-integer-to-nat, 11
- tcp, 8, 10, 31, 32, 37, 38, 40–43, 46, 48–54, 56, 59, 63, 65, 66
- tcp-add, 31
- tcp-divide, 38
- tcp-long-new-r, 54
- tcp-minus-twoto, 32
- tcp-mult, 37
- tcp-new-r, 40
- tcp-tc-minus, 32
- tcp-tc-to-integer, 10
- tcp-twoto, 32
- the-proof, 28
- the-proof-part2, 28
- the-proof-part3, 29
- times-1, 2
- times-2-not-1, 3
- times-2-twoto, 11
- times-add1, 2
- times-conserves-lessp, 66
- times-conserves-lessp-dual, 66
- times-conserves-lessp-dual-real, 66
- times-conserves-lessp-real, 66
- times-difference, 37
- times-distributes-over-remainder, 5
- times-quotient-lessp-relate, 40
- times-quotient-lessp-relate-dual, 40
- times-times-induct, 42
- times-to-mult-1, 38
- times-zero2, 2
- top-bit-off-implies-smaller, 31
- transfer-add, 46
- truep-boolp, 6
- twoto, 3, 9–13, 30–34, 36–41, 43, 44, 47–54, 57–59, 63, 66
- twoto-by-0, 3
- twoto-never-0, 3
- twoto-plus, 3
- unfold-negative-tc-to-integer, 43
- unfold-positive-tc-to-integer, 43
- up, 17, 20, 21
- up-shift-0, 36
- up-shift-1, 35–39, 55, 56, 60, 62, 64
- up-shift-1-append, 64
- up-shift-1-bv-vec, 65
- up-shift-1-invert-lsb, 39
- up-shift-n, 35, 36, 61, 65
- up-shift-strict, 65
- up-shift-strict-2, 65
- upper-bound-on-bv-to-nat, 10
- upper-bound-on-non-negative-bv-to-nat, 10
- us1-dos-1, 64
- vec-append, 36
- zero-bitvec, 7, 61–63, 65, 66
- zero-bitvec-extension, 65