— CVC4 proofs with bit-vectors (for the veriT proof format) —

Contents

1 Proof Calculus

The proof calculus is extended with rules for representing the bit-blasting of bit-vector terms. This is independent of other rules for preprocessing, Boolean and theory reasoning.

1.1 Bit-blasting

Judgment for bit-blasting:

$$\texttt{bbT}\ n\ x\ [x_0; \ldots; x_{n-1}]$$

in which n is a positive number, x is a term of the bit-vector type for size n, and x_i, with $x_i \in \{\bot, \top\}$, for $0 \leq i < n$, is the Boolean equivalent of the i-th bit of x (from least to most significant). Thus $[x_0; \ldots; x_{n-1}]$ is the Boolean bit-level interpretation corresponding to the bit-vector term x.

List of rules for bit-blasting

1.1.1 Variables and constants

$$\frac{}{\texttt{bbT}\ n\ x\ [x_0; \ldots; x_{n-1}]}\ \textsc{BbVar} \qquad \frac{}{\texttt{bbT}\ n\ v\ [v_0; \ldots; v_{n-1}]}\ \textsc{BbConst}$$

1.1.2 Bit-wise operations (bvand, bvor, bvxor, bvnot)

$$\frac{\texttt{bbT}\ n\ x\ [x_0;\ldots;x_{n-1}] \quad \texttt{bbT}\ n\ y\ [y_0;\ldots;y_{n-1}]}{\texttt{bbT}\ n\ (\mathit{bvand}\ x\ y)\ [x_0 \wedge y_0;\ldots;x_{n-1} \wedge y_{n-1}]}\ \textsc{BbAnd}$$

$$\frac{\texttt{bbT}\ n\ x\ [x_0;\ldots;x_{n-1}] \quad \texttt{bbT}\ n\ y\ [y_0;\ldots;y_{n-1}]}{\texttt{bbT}\ n\ (\mathit{bvor}\ x\ y)\ [x_0 \vee y_0;\ldots;x_{n-1} \vee y_{n-1}]}\ \textsc{BbOr}$$

$$\frac{\texttt{bbT}\ n\ x\ [x_0;\ldots;x_{n-1}] \quad \texttt{bbT}\ n\ y\ [y_0;\ldots;y_{n-1}]}{\texttt{bbT}\ n\ (\mathit{bvxor}\ x\ y)\ [x_0 \oplus y_0;\ldots;x_{n-1} \oplus y_{n-1}]}\ \textsc{BbXor}$$

$$\frac{\texttt{bbT}\ n\ x\ [x_0;\ldots;x_{n-1}]}{\texttt{bbT}\ n\ (\mathit{bvnot}\ x)\ [\neg x_0;\ldots;\neg x_{n-1}]}\ \textsc{BbNot}$$

1.1.3 Arithmetic operations (addition, negation, multiplication)

Addition Bit-blasted following the *ripple carry adder* way of computing:

$$\frac{\texttt{bbT}\ n\ x\ [x_0;\ldots;x_{n-1}] \quad \texttt{bbT}\ n\ y\ [y_0;\ldots;y_{n-1}]}{\texttt{bbT}\ n\ (\mathit{bvadd}\ x\ y)\ [(x_0 \oplus y_0) \oplus \mathrm{carry}_0;\ldots;(x_{n-1} \oplus y_{n-1}) \oplus \mathrm{carry}_{n-1}]}\ \textsc{BbAdd}$$

in which, for $i \geq 0$,

$$\begin{array}{rcl} \mathrm{carry}_0 & = & \bot \\ \mathrm{carry}_{i+1} & = & (x_i \wedge y_i) \vee ((x_i \oplus y_i) \wedge \mathrm{carry}_i) \end{array}$$

Negation Bit-blasted following $(\mathit{bvneg}\ a) \equiv (\mathit{bvadd}\ (\mathit{bvnot}\ a)\ 1)$:

$$\frac{\texttt{bbT}\ n\ x\ [x_0;\ldots;x_{n-1}]}{\texttt{bbT}\ n\ (\mathit{bvneg}\ x)\ [(\neg x_0 \oplus \bot) \oplus \mathrm{carry}_0;\ldots;(\neg x_{n-1} \oplus \bot) \oplus \mathrm{carry}_{n-1}]}\ \textsc{BbNeg}$$

in which, for $i \geq 0$,

$$\begin{array}{rcl} \mathrm{carry}_0 & = & \top \\ \mathrm{carry}_{i+1} & = & (\neg x_i \wedge \bot) \vee ((\neg x_i \oplus \bot) \wedge \mathrm{carry}_i) \end{array}$$

Multiplication Bit-blasted following the *shift add multiplier* way of computing:

$$\frac{\texttt{bbT}\ n\ x\ [x_0;\ldots;x_{n-1}] \quad \texttt{bbT}\ n\ y\ [y_0;\ldots;y_{n-1}]}{\texttt{bbT}\ n\ (\mathit{bvmult}\ x\ y)\ [\mathrm{res}_0^{n-1};\ldots;\mathrm{res}_{n-1}^{n-1}]}\ \textsc{BbMult}$$

in which, for $i, j \geq 0$,

$$
\begin{array}{lcl}
\mathrm{res}_i^0 & = & \mathrm{sh}_i^0 \\
\mathrm{res}_0^j & = & \mathrm{sh}_0^0 \\
\mathrm{res}_i^{j+1} & = & \mathrm{res}_i^j \oplus \mathrm{sh}_i^{j+1} \oplus \mathrm{carry}_i^{j+1} \\
\mathrm{carry}_i^0 & = & \bot \\
\mathrm{carry}_{i+1}^{j+1} & = & \left\{ \begin{array}{ll} \mathrm{res}_i^j \wedge \mathrm{sh}_i^{j+1} \vee ((\mathrm{res}_i^j \oplus \mathrm{sh}_i^{j+1}) \wedge \mathrm{carry}_i^{j+1}) & \text{if } j < i \\ \bot & \text{otherwise} \end{array} \right. \\
\mathrm{sh}_i^j & = & \left\{ \begin{array}{ll} x_{i-j} \wedge y_j & \text{if } j \leq i \\ \bot & \text{otherwise} \end{array} \right.
\end{array}
$$

Example mult for $n = 4$:

$$
\left[
\begin{array}{lcl}
\mathrm{res}_0^3 & = & (x_0 \wedge y_0); \\
\mathrm{res}_1^3 & = & \underbrace{(x_1 \wedge y_0) \oplus (x_0 \wedge y_1)}_{\underbrace{\mathrm{res}_1^0 \oplus \mathrm{sh}_1^1}_{\mathrm{res}_1^2}} \oplus \underbrace{\mathrm{carry}_1^3}_{\bot}; \\
\mathrm{res}_2^3 & = & \underbrace{\underbrace{(x_2 \wedge y_0) \oplus (x_1 \wedge y_1)}_{\underbrace{\mathrm{res}_2^0 \oplus \mathrm{sh}_2^1}_{\mathrm{res}_2^1}} \oplus \underbrace{\mathrm{carry}_2^1}_{\mathrm{res}_1^0 \wedge \mathrm{sh}_1^1 \ldots}}_{\mathrm{res}_2^2} \oplus \underbrace{(x_0 \wedge y_2)}_{\mathrm{sh}_2^2} \oplus \underbrace{\mathrm{carry}_2^3}_{\bot}; \\
\mathrm{res}_3^3 & = & \underbrace{\underbrace{\underbrace{(x_3 \wedge y_0) \oplus (x_2 \wedge y_1)}_{\mathrm{res}_3^0 \oplus \mathrm{sh}_3^1} \oplus \underbrace{\mathrm{carry}_3^1}_{\ldots}}_{\mathrm{res}_3^1} \oplus \underbrace{(x_1 \wedge y_2)}_{\mathrm{sh}_3^2} \oplus \underbrace{\mathrm{carry}_3^2}_{\mathrm{res}_2^1 \wedge \mathrm{sh}_2^2 \ldots}}_{\mathrm{res}_3^2} \oplus \underbrace{(x_0 \wedge y_3)}_{\mathrm{sh}_3^3} \oplus \underbrace{\mathrm{carry}_3^3}_{\bot};
\end{array}
\right]
$$

with

$$
\begin{array}{lcl}
\mathrm{carry}_3^1 & = & \mathrm{res}_2^0 \wedge \mathrm{sh}_2^1 \vee ((\mathrm{res}_2^0 \oplus \mathrm{sh}_2^1) \wedge \mathrm{carry}_2^1) \\
\mathrm{carry}_2^1 & = & \mathrm{res}_1^0 \wedge \mathrm{sh}_1^1 \vee ((\mathrm{res}_1^0 \oplus \mathrm{sh}_1^1) \wedge \underbrace{\mathrm{carry}_1^1}_{\bot}) \\
\mathrm{carry}_3^2 & = & \mathrm{res}_2^1 \wedge \mathrm{sh}_2^2 \vee ((\mathrm{res}_2^1 \oplus \mathrm{sh}_2^2) \wedge \underbrace{\mathrm{carry}_2^2}_{\bot})
\end{array}
$$

1.1.4 Extraction

$$
\frac{\texttt{bbT}\ n\ x\ [x_0; \ldots; x_{n-1}] \quad 0 \leq j \leq i < n}{\texttt{bbT}\ i - j + 1\ (\mathit{extract}\ i\ j\ x)\ [x_j; \ldots; x_i]}\ \textsc{BbExtract}
$$

1.1.5 Concatenation

$$\frac{\texttt{bbT}\ n\ x\ [x_0; \ldots; x_{n-1}] \quad \texttt{bbT}\ m\ y\ [y_0; \ldots; y_{m-1}]}{\texttt{bbT}\ n+m\ (concat\ x\ y)\ [y_0; \ldots; y_{m-1}; x_0; \ldots; x_{n-1}]}\ \textsc{BbConcat}$$

1.1.6 Equality

$$\frac{\texttt{bbT}\ n\ x\ [x_0; \ldots; x_{n-1}] \quad \texttt{bbT}\ n\ y\ [y_0; \ldots; y_{n-1}]}{(bveq\ x\ y)\ \leftrightarrow\ (x_0 \leftrightarrow y_0\ \wedge\ \ldots\ \wedge\ x_{n-1} \leftrightarrow y_{n-1})}\ \textsc{BbEq}$$

1.1.7 Comparison predicates (signed/unsigned)

Unsigned less than Bit-blasted following $a < b$ iff, for $0 \leq i < n$, $(\sim a[i]$ AND $b[i])$ OR $(a[i] \leftrightarrow b[i]$ AND $a[i+1 : n] < b[i+1 : n])$

$$\frac{\texttt{bbT}\ n\ x\ [x_0; \ldots; x_{n-1}] \quad \texttt{bbT}\ n\ y\ [y_0; \ldots; y_{n-1}]}{(bvult\ x\ y)\ \leftrightarrow\ \bigvee_{i=0}^{n-1} \mathrm{res}_{(n-1)-i}}\ \textsc{BbUlt}$$

in which, for $i \geq 0$,

$$\begin{array}{rcl} \mathrm{res}_0 & = & \neg x_0 \wedge y_0 \\ \mathrm{res}_{i+1} & = & ((x_{i+1} \leftrightarrow y_{i+1}) \wedge \mathrm{res}_i) \vee (\neg x_{i+1} \wedge y_{i+1}) \end{array}$$

Signed less than Bit-blasted following $a < b$ iff a is negative and b positive, or they have the same sign and the value of a is less than the value of b (i.e. the same as above)

$$\frac{\texttt{bbT}\ n\ x\ [x_0; \ldots; x_{n-1}] \quad \texttt{bbT}\ n\ y\ [y_0; \ldots; y_{n-1}]}{(bvslt\ x\ y)\ \leftrightarrow\ (x_{n-1} \wedge \neg y_{n-1}) \vee \Big((x_{n-1} \wedge y_{n-1}) \wedge \bigvee_{i=0}^{n-2} \mathrm{res}_{(n-2)-i}\Big)}\ \textsc{BbSlt}$$

in which, for $i \geq 0$,

$$\begin{array}{rcl} \mathrm{res}_0 & = & \neg x_0 \wedge y_0 \\ \mathrm{res}_{i+1} & = & ((x_{i+1} \leftrightarrow y_{i+1}) \wedge \mathrm{res}_i) \vee (\neg x_{i+1} \wedge y_{i+1}) \end{array}$$

1.1.8 Signed extension

$$\frac{\texttt{bbT}\ n\ x\ [x_0; \ldots; x_{n-1}]}{\texttt{bbT}\ n+k\ (bbsextend\ k\ x)\ [x_0; \ldots; x_{n-1}; x_n; \ldots; x_{n+k-1}]}\ \textsc{BbSExt}$$

2 veriT proof format

Proofs involving bit-vector reasoning will have an overall structure with four components:

1. Inputs
2. Definition of bit-vector terms (via the above bit-blasting rules)
3. Resolution subproof of each bit-vector lemma learned

   The subproof will depend on the bit-blasting definitions and Boolean reasoning, such that one can derive e.g. the following valid clause

```
(resolution ((not (= #b1 a)) (not (= #b0 b)) (not (= #b1 c)) (= (bvadd a b) c)) n1...nk)
```

in which `n1`, ..., `nk` will be the Boolean reasoning clauses.

4. Resolution proof using the learned bit-vector lemmas and inputs to derive a conflict.

2.1 Bit-blasting

Examples of BbVar and BbConst:

```
(bbvar ((bbT a [ (bitof 0 a) (bitof 1 a) (bitof 2 a) (bitof 3 a)])))
(bbconst ((bbT #b0010 [ false true false false])))
```

Example of BbEq:

```
(bbeq ((=
         (= #b0010 a)
         (and
           (= false (bitof 0 a))
           (and
             (= true (bitof 1 a))
             (and
               (= false (bitof 2 a))
               (= false (bitof 3 a))))))) n1 n2)
```

in which `n1` is the definition of `#b0010` via BbConst and `n2` the definition of `a` via BbVar.