New Theory for Verilog Size Checking

Path

A path is represented as a list of natural numbers that encodes navigation through the abstract syntax tree: starting from the root, each number indicates which child to visit next (with 0 denoting the first child, 1 the second, and so forth). For example, the path $[1, 0]$ refers to the first child of the second child of the root expression. The empty path $[]$ refers to the root expression itself.

Expressions

- $\mathcal{A}$ is the set of atoms in an expression. In the AST, they correspond to leaves. In System-Verilog, they refer to what the standard calls an "operand" (variables, integers, function calls, slices of a variable, etc.).
- $\mathcal{R}$ is the set of resizable expressions. It corresponds to the expression whose top level is one of: atom, comparisons, logic operation, reduction, assignments, shift assignments, concatenation, replication and inside operation.
- $\mathcal{E}$ is the set of System-Verilog expressions. We have $\mathcal{A} \subset \mathcal{E}$. An expression either contains an expression or is an atom.

Rules

We use the following notations:

- Γ maps atoms to their size,
- Φ maps lvalues to their size,
- The statement $e \Rightarrow t \dashv f$ means "e has size t, with f mapping each sub-expressions of e to their size",
- The statement $e \Leftarrow t \dashv f$ means "e can be resized to t, with f mapping each sub-expressions of e to their size".

Function combinator

$$\mathrm{Unary}(f, t) ::= \begin{cases} [] & \mapsto \quad t \\ 0 :: p & \mapsto \quad f(p) \end{cases} \qquad \mathrm{Binary}(t, f, g) ::= \begin{cases} [] & \mapsto \quad t \\ 0 :: p & \mapsto \quad f(p) \\ 1 :: p & \mapsto \quad g(p) \end{cases}$$

$$\mathrm{Ternary}(t, f, g, h) ::= \begin{cases} [] & \mapsto \quad t \\ 0 :: p & \mapsto \quad f(p) \\ 1 :: p & \mapsto \quad g(p) \\ 2 :: p & \mapsto \quad h(p) \end{cases} \qquad \mathrm{Nary}(t, f_1, \ldots, f_k) ::= \begin{cases} [] & \mapsto \quad t \\ i :: p & \mapsto \quad f_i(p) \end{cases}$$

Resizing

$$\frac{e \Rightarrow s \dashv f \qquad s \leqslant t \qquad e \in \mathcal{R}}{e \Leftarrow t \dashv f\left[[] \mapsto t\right]}\ \textsf{Resize}{\Leftarrow}$$

H

$$\frac{a \Leftarrow t \dashv f_a \qquad b \Leftarrow t \dashv f_b}{a \oplus b \Leftarrow t \dashv \mathrm{Binary}(t, f_a, f_b)}\ \textsf{BinOp}{\Leftarrow} \qquad \frac{a \Leftarrow t \dashv f_a \qquad b \Rightarrow t_b \dashv f_b}{a \oplus b \Leftarrow t \dashv \mathrm{Binary}(t, f_a, f_b)}\ \textsf{Shift}{\Leftarrow}$$

$$\frac{e \Leftarrow t \dashv f}{\oplus e \Leftarrow t \dashv \mathrm{Unary}(t, f)}\ \textsf{UnOp}{\Leftarrow} \qquad \frac{e \Rightarrow t_e \dashv f_e \qquad a \Leftarrow t \dashv f_a \qquad b \Leftarrow t \dashv f_b}{e ? a : b \Leftarrow t \dashv \mathrm{Ternary}(t, f_e, f_a, f_b)}\ \textsf{Cond}{\Leftarrow}$$

Synthesize

$$\frac{\Gamma(e) = s \qquad e \in \mathcal{O}}{e \Rightarrow s \dashv \{[] \mapsto s\}} \ \textsf{Operand}{\Rightarrow}$$

$$\frac{e \Rightarrow t \dashv f}{\oplus e \Rightarrow t \dashv \text{Unary}(t, f)} \ \textsf{UnOp}{\Rightarrow} \qquad \frac{e \Rightarrow t \dashv f}{\oplus e \Rightarrow 1 \dashv \text{Unary}(1, f)} \ \textsf{Red}{\Rightarrow}$$

$$\frac{a \Rightarrow t \dashv f_a \qquad b \Rightarrow t_b \dashv f_b}{a \oplus b \Rightarrow t \dashv \text{Binary}(t, f_a, f_b)} \ \textsf{Shift}{\Rightarrow} \qquad \frac{a \Rightarrow t_a \dashv f_a \qquad b \Rightarrow t_b \dashv f_b}{a \oplus b \Rightarrow 1 \dashv \text{Binary}(1, f_a, f_b)} \ \textsf{Logic}{\Rightarrow}$$

$$\frac{a \Rightarrow t \dashv f_a \qquad b \Leftarrow t \dashv f_b}{a \oplus b \Rightarrow t \dashv \text{Binary}(t, f_a, f_b)} \ \textsf{LBinOp}{\Rightarrow} \qquad \frac{a \Leftarrow t \dashv f_a \qquad b \Rightarrow t \dashv f_b}{a \oplus b \Rightarrow t \dashv \text{Binary}(t, f_a, f_b)} \ \textsf{RBinOp}{\Rightarrow}$$

$$\frac{a \Rightarrow t \dashv f_a \qquad b \Leftarrow t \dashv f_b}{a \oplus b \Rightarrow 1 \dashv \text{Binary}(1, f_a, f_b)} \ \textsf{LCmp}{\Rightarrow} \qquad \frac{a \Leftarrow t \dashv f_a \qquad b \Rightarrow t \dashv f_b}{a \oplus b \Rightarrow 1 \dashv \text{Binary}(1, f_a, f_b)} \ \textsf{RCmp}{\Rightarrow}$$

$$\frac{\phi(l) = t \qquad e \Leftarrow t \dashv f}{(l = e) \Rightarrow t \dashv \text{Unary}(t, f)} \ \textsf{LAssign}{\Rightarrow} \qquad \frac{\phi(l) = t \qquad e \Rightarrow t_e \dashv f \qquad t < t_e}{(l = e) \Rightarrow t \dashv \text{Unary}(t, f)} \ \textsf{RAssign}{\Rightarrow}$$

$$\frac{e \Rightarrow t_e \dashv f_e \qquad a \Rightarrow t \dashv f_a \qquad b \Leftarrow t \dashv f_b}{e?a\!:\!b \Rightarrow t \dashv \text{Ternary}(t, f_e, f_a, f_b)} \ \textsf{LCond}{\Rightarrow}$$

$$\frac{e \Rightarrow t_e \dashv f_e \qquad a \Leftarrow t \dashv f_a \qquad b \Rightarrow t \dashv f_b}{e?a\!:\!b \Rightarrow t \dashv \text{Ternary}(t, f_e, f_a, f_b)} \ \textsf{RCond}{\Rightarrow}$$

$$\frac{i \in \mathbb{N} \qquad e \Rightarrow t_e \dashv f \qquad t = i \times t_e}{\{i\ e\} \Rightarrow t \dashv \text{Unary}(t, f)} \ \textsf{Repl}{\Rightarrow}$$

$$\frac{e_1 \Rightarrow t_1 \dashv f_i \qquad \dots \qquad e_k \Rightarrow t_k \dashv f_k \qquad t = t_1 + \cdots + t_k}{\{e_1, \dots, e_k\} \Rightarrow t \dashv \text{Nary}(t, f_1, \dots, f_k)} \ \textsf{Concat}{\Rightarrow}$$