FEEDBACK

4.2.3.2 Foundation Language (FL) operators

Table 2 is not referenced in the text, yet the table is presented.

4.2.3.2.11 Suffix implication operators

For any flavor of PSL, the FL operators with the next highest precedence are those used to describe behavior
consisting of a property that holds at the end of a given sequence. These operators are:

`|->` overlapping suffix implication
`|=>` non-overlapping suffix implication

Ben: I suggest that we delete the word "suffix" in the above definition to be in line with SystemVerilog. Thus,

|-> overlapping implication
|=> non-overlapping implication

--- COMMENT
The LRM makes no reference to the word "antecedent" or "consequent". Those are terms that are common in the industry. From SystemVerilog LRM:

The result of the implication is either true or false. The left-hand side operand *sequence_expr* is called the *antecedent*, while the right-hand side operand *property_expr* is called the *consequent*.

6.1.1.1.2 SERE fusion (:)

Informal Semantics

For SEREs `A` and `B`:

`A:B` holds tightly on a path iff there is a future cycle *n*, such that `A` holds tightly on the path up to and including the *n*th cycle and `B` holds tightly on the path starting at the *n*th cycle.

Ben: I suggest that we add a comment that provides additional clarification, as shown in the SystemVerilog LRM.

An *empty sequence* is one that does not match over any positive number of clocks. The following rules apply for concatenating sequences with empty sequences. An empty sequence is denoted as *empty* and a sequence is denoted as *seq*.

— (*empty* `:` *seq*) does not result in a match
— (*seq* `:` *empty*) does not result in a match
— (*empty* ; `[*n];` *seq*), where n is greater than 0, is equivalent to ([*n-1]; seq)
— (*seq*; [*`n];` *empty*), where n is greater than 0, is equivalent to (seq ; [*n-1])

For example,

```
{b ;  {a[*0] : c}}
```

produces no match of the sequence.

```
{b ; a[*0:1]; [*2]; c)
```

is equivalent to

```
{b ; [*2]; c} | {b ; a; [*2]; c}
```