F.2.3.6 Derived sampled value functions

ADD to the end

— `$rose_gclk(`e`)` ≡ `$past_gclk(`b`) !== 1 && `b` === 1`, where b is the LSB of e.

— `$fell_gclk(`e`)` ≡ `$past_gclk(`b`) !== 0 && `b` === 0`, where b is the LSB of e.

— `$stable_gclk(`e`)` ≡ `$past_gclk(`e`) === `e.

— `$changed_gclk(`e`)` ≡ `$past_gclk(`e`) !== `e.

— `$rising_gclk(`e`)` ≡ b `!== 1 && $future_gclk(`b`) === 1`, where b is the LSB of e.

— `$falling_gclk(`e`)` ≡ b `!== 0 && $future_gclk(`b`) === 0`, where b is the LSB of e.

— `$steady_gclk(`e`)` ≡ e `=== $future_gclk(`e`).`

— `$changing_gclk(`e`)` ≡ e `!== $future_gclk(`e`).`

F.4 Extended expressions

F.4.1 Extended booleans

REPLACE

This subclause describes the semantics of several constructs that are used like expressions, but whose meaning at a point in a word can depend both on the letter at that point and on previous letters in the word. By abuse of notation, the meanings of these extended expressions are defined for letters denoted "w^j" even though they depend also on letters w^i for $i \leq j$. The reason for this abuse is to make clear the way these definitions should be used in combination with those in preceding subclauses.

WITH

This subclause describes the semantics of several constructs that are used like expressions, but whose meaning at a point in a word ~~can~~ may depend both on the letter at that point and on ~~previous~~ other letters in the word. By abuse of notation, the meanings of these extended expressions are defined for letters denoted "w^j" even though they depend also on letters w^i for $i \leq \neq j$. The reason for this abuse is to make clear the way these definitions should be used in combination with those in preceding subclauses.

F.4.2 Past

REPLACE

Let $n \geq 1$. If there exists $0 \leq i < j$ so that
$w^{i,j}, \{\}, \{\} \models$ `((c &&` e_2`) ##1 (c &&` e_2`[=n-1] ##1 1)`, then `$past(`e_1`, n,` e_2`, c)`$[w^j] = e_1[w^i]$.

Otherwise, `$past(`e_1`, n, `e_2`, c)`$[w^j]$ is the result of evaluating the expression e_1 using the initial values of the variables comprising the expression. The initial value of a static variable is the value assigned in its declaration, or, in the absence of such an assignment, it is the default (or uninitialized) value of the corresponding type (see 6.7, Table 6-1). The initial value of any other variable or signal is the default value of the corresponding type (see 6.7, Table 6-1).

WITH

— Let $n \geq 1$. If there exists $0 \leq i < j$ so that
$w^{i,j}, \{\}, \{\} \models$ `((c && `e_2`) ##1 (c && `e_2`[=n-1] ##1 1)`,
then `$past(`e_1`, n, `e_2`, c)`$[w^j] = e_1[w^i]$. Otherwise, `$past(`e_1`, n, `e_2`, c)`$[w^j]$ is the result of evaluating the expression e_1 using the initial values of the variables comprising the expression. The initial value of a static variable is the value assigned in its declaration, or, in the absence of such an assignment, it is the default (or uninitialized) value of the corresponding type (see 6.7, Table 6-1). The initial value of any other variable or signal is the default value of the corresponding type (see 6.7, Table 6-1).

— If $j > 0$ then `$past_gclk(`e`)`$[w^j] = e[w^{j-1}]$. `$past_gclk(`e`)`$[w^0]$ is the result of evaluating the expression e using the initial values of the variables comprising the expression.

Note. that `$past_gclk(`e`)` is equivalent to `$past(`e`,1,1'b1,1'b1)`.

ADD

F.4.3 Future

w denotes a nonempty finite or infinite word over Σ, and j denotes an integer so that $0 \leq j < |w| - 1$. `$future_gclk(`e`)`$[w^j] = e[w^{j+1}]$. If w is a finite word, `$future_gclk(`e`)`$[w^{|w|-1}]$ is undefined.