📖 **Rule**: The use of **`$sampled`** in assertions, although allowed, is redundant because the values used for all design variables inside the expressions are those sampled at the Preponed region. However, the **`$sampled`** is necessary in an action block, which does not follow/utilize the sampled semantics and hence uses the current values of the variables at the time the action block is evaluated.

4.2.1.1.2 $past

📖 **Rule**: The **`$past`** function provides the sampled value that an expression held in a previous n[th] cycle. The syntax of the function is: *[1]*

```
$past( expression1 [, number_of_ticks] [, expression2] [, clocking_event])
```

`expression1` represents the expression being sought.
The three optional arguments define the following:

- `expression1` and `expression2` can be any expression allowed in assertions.
- `number_of_ticks` specifies the number of clock ticks in the past. `number_of_ticks` must be one or greater, and must be static (i.e., known at elaboration time). If `number_of_ticks` is not specified, then it defaults to 1. If the specified clock tick in the past is before the start of simulation, the returned value from the **`$past`** function is a value of `X`.
- `expression2` is used as a gating expression for the clocking event. The value returned for **`$past`** is `expression1` sampled `number_of_ticks` gated cycles ago. In other words, for: **`$past`**`(data, 3, load_enable, @(`**`posedge`**` clk))` the returned value is the sampled value of `data` in the 3rd prior cycle in which `load_enable` was true. This is demonstrated in Figure 4.1.1.1-2 ***/ch4/4.2/past.sv***
- `clocking_event` specifies the clocking event for sampling expression. A clock tick is based on `clocking_event`.
- Examples: :

```
regload |=> reg_data==$past(data);      // value of load_data at the previous cycle
regload |-> ##2 reg_data==$past(data, 2);  // value of load_data at 2 cycles ago
regload |-> ##2 reg_data==$past(data, 2, 1, @(posedge clk)); // value of load_data at 2 cycles ago
regload |-> ##2 reg_data==$past(data, 3, load_enable, @ (posedge clk) );
// value of data when it was sampled 3 gated cycles ago with load_enable as the gate.
```

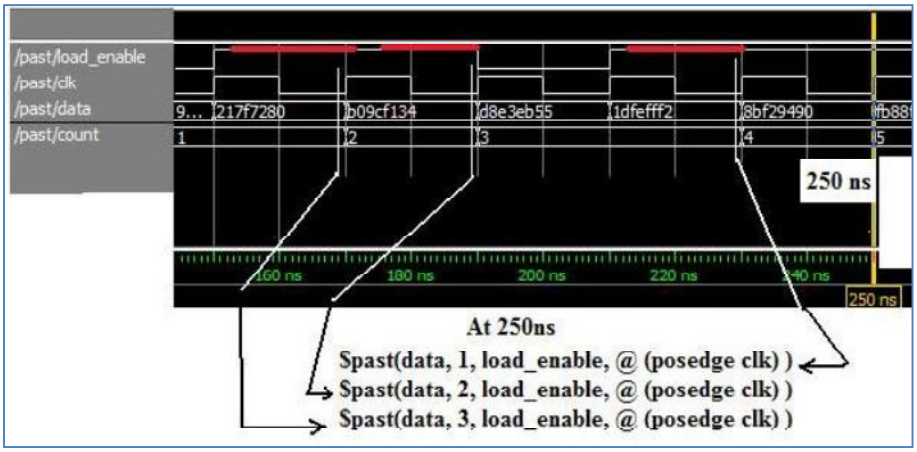

Figure 4.2.1.1-2 Evaluation of `$past(data, 3, load_enable, @ (posedge clk)`

📖 **Rule**: Method **`triggered`** (see 2.5.2**)** is not allowed as an argument of system task functions. In addition, it can only be used within a SVA construct. Thus, the following code is illegal:

```
sequence qT; @ (posedge clk)a ##2 b; endsequence : qT
a_P1 : assert property (@ (posedge clk)  go |=> $past(qT.triggered));  // 💣
wire go_triggered;
assign go_triggered = $past(qT.triggered));  // 💣    ch4/ sampled4_3.sv
```