Assertion API errata update as of 7-Jan-2004

[all references related to SV 3.1a draft 2,
and assume an earlier application of the changes defined in LRM 121]

Section 28.4.2, "Placing assertion callbacks"

update this part of LRM-121 [related to draft2 paper page 341, prototype for vpi_register_assertion_cb()]:

Change (changes in ~~red~~ and blue):

Use `vpi_register_assertion_cb()` to place an assertion callback, the prototype is:

```
/* typedef for vpi_register_assertion_cb callback function */
typedef  PLI_INT32 (vpi_assertion_callback_func)(
   PLI_INT32 reason,          /* callback reason */
   p_vpi_time cb_time,        /* callback time */
   vpiHandle assertion,       /* handle to assertion */
   p_vpi_attempt_info info,   /* attempt related information */
   PLI_BYTE8 *user_data       /* user data entered upon registration */
);

vpiHandle vpi_register_assertion_cb(
      vpiHandle assertion,       /* handle to assertion */
      PLI_INT32 reason,          /* reason for which callbacks needed */
```
~~PLI_INT32 (*cb_rtn)(/* callback function */~~
~~PLI_INT32 reason, /* callback reason */~~
~~p_vpi_time cb_time, /* callback time */~~
~~vpiHandle assertion,~~
~~p_vpi_attempt_info info,~~
~~PLI_BYTE8 *userData),~~
```
      vpi_assertion_callback_func *cb_rtn,
      PLI_BYTE8 *user_data       /* user data to be supplied to cb */
);

typedef struct t_vpi_assertion_step_info {
      PLI_INT32 matched_expression_count;
      vpiHandle *matched_exprs;                /* array of expressions */
      p_vpi_source_info *exprs_source_info;    /* array of source info */
      PLI_INT32 stateFrom, stateTo;            /* identify transition */
} s_vpi_assertion_step_info, *p_vpi_assertion_step_info;

typedef struct t_vpi_attempt_info {
      union {
            vpiHandle failExpr;
            p_vpi_assertion_step_info step;
      } detail;
      s_vpi_time attemptStartTime; /* Time attempt triggered */
} s_vpi_attempt_info, *p_vpi_attempt_info;
```

[towards the end of the section (draft2: at paper page 342), make the following changes (in Blue), also add the 3rd and 4th NOTES at the end of this section]

The `attempt information structure` contains details relevant to the specific event that occurred.

— On disable, enable, reset and kill callbacks, the `info` field is NULL.

— On start and success callbacks, only the `attemptStartTime` field is valid.

— On a `cbAssertionFailure` callback, the `attemptStartTime` and `detail.failExpr` fields are valid.

— On a step callback, the `attemptStartTime` and `detail.step` elements ~~elements~~fields are valid.

NOTES

1) ...

2) ...

3) The content of the `cb_time` field depends on the reason identified by the reason field, as follows:
— `cbAssertionStart` - `cb_time` is the time when the assertion attempt has been started.
— `cbAssertionSuccess`, `cbAssertionFailure` - `cb_time` is the time when the assertion succeeded/failed.
— `cbAssertionStepSuccess`, `cbAssertionStepFailure` - `cb_time` is the time when the assertion attempt step succeeded/failed.
— `cbAssertionDisable`, `cbAssertionEnable`, `cbAssertionReset`, `cbAssertionKill` - not possible, data supplied is NULL.

4) In contrast to `cb_time`, the content of `attemptStartTime` is *always* the start time of the actual attempt of an assertion. It can be used as an unique ID that distinguishes the attempts of any given assertion.