RELEASE_NOTES: SpecC Language Reference Manual, Version 2.0 (October 31, 2002)

This SpecC Language Reference Manual (LRM) version 2.0 is based on the previous version, SpecC LRM V1.0, dated March 6, 2001.

The following modifications have been applied:

* the abstract, the introduction and the references have been extended and updated

* the list of members of the language specification working group (LS-WG) has been added to the contributors section

* a new appendix B has been added with a description of the standard SpecC library; for now, the library consists of:
  - API to the simulation engine (which always was there but never had been documented so far)
  - SpecC standard channels (newly added as discussed under synchronization; see below)

* in the SpecC language definition (chapter 2), many sections have been clearified and updated, some were completely rewritten; also, the writing style and the English have been improved; finally, the examples have been fixed so that they actually contain valid SpecC code

* the SpecC syntax and grammar have been updated to reflect all changes introduced with SpecC version 2.0; in particular, the following new keywords have been added:
  - 'fsmd' (see section 2.4.5)
  - 'signal' (see section 2.2.6)
  - 'buffered' (see section 2.2.7)
  - 'rising' (see sections 2.2.6, 2.4.5)
  - 'falling' (see sections 2.2.6, 2.4.5)
  - 'fix' (reserved for future fixed-point data type, section A.1.5)

* Atomicity / Preemptive execution semantics
  [voting item 1 / discussion item 1.1]
  - chapter 3 about SpecC execution semantics based on time interval formalism has been added; also, several notes have been added to appropriate sections that there is no atomicity guarantee for any SpecC execution and preemptive execution is possible

* Mutual exclusion
  [voting item 2 / discussion item 1.2]
  - the mutual exclusive execution of code in channels by use of an implicit mutex in each channel instance has been added (see section 2.3.2, rule (j) and notes viii. through xiii.)

* Synchronization semantics
  [voting item 3 / discussion item 1.3]
  - chapter 3 defining the time interval formalism has been added
  - the new section 3.5 addresses synchronization in SpecC execution semantics
  - in addition, section 3.6 describes an abstract simulation algorithm
  - finally, appendix B.2 has been added with a description of the SpecC standard channel library

* Exception handlers
  [voting item 4 / discussion item 1.4]
  - the description of exception handling has been modified to allow exceptions to occur at any time and place, non-deterministically chosen (see section 2.4.7, rule (d) and note i.)

* Fixed-point data type
  [voting item 5 / discussion item 2.1]
  - since the introduction of a fixed-point data type has been agreed upon but is postponed for a later version of SpecC, only the new keyword 'fix' has been added to the reserved keywords in section A.1.5

* Distinguishing notified events
  [voting item 6 / discussion item 2.2]
  - although no real change was required, an explaining sentence has been added to the notes in Section 2.4.6

* AND conditions for event wait statement
  [voting item 7 / discussion item 2.3]
  - extended the SpecC grammar to allow AND operator for 'wait' statements
  - for consistency, the OR operator has been added also for 'wait' statements
    (with the same semantics as the old comma operator)
  - added/modified the rules for the semantics of 'wait' and added
    two explaining notes (see rules 2.4.6 (d) and (e), and notes i. and ii.)

* Simplify the notation of par/pipe/try statement
  [voting item 8 / discussion item 2.4]
  - a short cut for the notation of behavior 'main' calls has been added
    to the rules for behaviors (see rule 2.3.1(k))
  - now the statement "b;" is equivalent to the statement "b.main();"
  - note that this feature does not require any change in the grammar
    because it syntactically falls into the grammar rules of expressions
  - as a consequence, these short-cuts are allowed anywhere in the code,
    not only within the 'par', 'pipe', 'try'-'trap'-'interrupt' constructs
  - in order to point out the usefulness of this short-cut, appropriate
    notes have been added to the descriptions of sequential execution and
    the 'par', 'pipe' and 'try'-'trap'-'interrupt' statements

* Local method function call in FSM statement
  [voting item 9 / discussion item 2.6]
  - extended the grammar of the 'fsm' construct such that a compound
    statement with local code is allowed between the state name and the colon
  - note that the original syntax described in the voting proposal is very
    ambiguous and clashes with regular compound statements; therefore,
    a different syntax/grammar needed to be selected
  - note that this syntax also has the advantage that it keeps the state code
    and state transistions clearly separated by the colon
  - the rules of the 'fsm' statement have been adjusted and extended in order
    to accomodate the option of local states (see section 2.4.4)

* Extension of persistent annotation
  [voting item 10 / discussion item 2.7]
  - extended the SpecC grammar to allow composite annotations
  - reworked, clarified and extended the description of annotations
    and the example accordingly (see section 2.5.2)

* Modeling RTL in SpecC
  [voting item 11 / discussion item 3.1]
  - added keywords and extended the SpecC grammar to support
    'signal' and 'buffered' data types, 'rising' and 'falling' operators
    on signals, and the 'fsmd' construct
  - added section 2.2.6 defining 'signal' data types, plus an example
  - added section 2.2.7 defining 'buffered' data types, plus an example
  - added section 2.4.5 defining the 'fsmd' construct, plus an example
  - note that some changes to the original voting proposal were necessary
    in order to have a non-ambiguous syntax and grammar
  - the keywords 'rising' and 'falling' have been introduced in order
    to avoid ambiguities regarding the sensitivity to value changes of signals
  - the implicit infinite loop of the 'fsmd' statement has been dropped
    because it is not necessary and would prohibit composite FSMDs
  - asynchronous reset has been refined to an explicit reset signal which
    also can be specified for 'buffered' variables (the original proposal
    using the 'trap' keyword was ambiguous and not implementable)

* finally, numerous small changes, adjustments and clarifications
  have been applied to the document

For any kind of feedback, bug reports or other suggestions,
the authors can be reached best by email:

  Rainer Doemer   <doemer@cecs.uci.edu>
  Andreas Gerstlauer <gerstl@cecs.uci.edu>
  Daniel Gajski   <gajski@cecs.uci.edu>

Best regards,

Rainer Doemer, November 1, 2002.