Dear Editor,

We have carefully read all the reviewer comments and made changes to the paper accordingly. We would like to thank the anonymous reviewers for providing detailed comments, which we ourselves think has significantly improved the manuscript in terms of readability and quality. We have also put in further effort to restructure the paper in order to make it accessible to a wider audience.

All the changes made following the reviewer comments are attached to this letter.

Regards,

Authors

# Reviewer 1

|  |  |  |
| --- | --- | --- |
|  | **Comments** | **Suggestions** |
|  | The presented idea seems attractive, and the paper is well-written. However, one issue that I think deserves further discussion is that, as you mention in Sec. 6.1, the programmer must be aware of the interaction of wait and other pause statements. Your example shows that S is not emitted each ms because of the extra pause. However, the introduction of additional (and in fact at design time unpredictable many) pause statements also changes the semantics of concurrent synchronous programs which comprise threads that communicate with each other.  This becomes also an interesting point in 7 where you discuss the differences between your approach and Bourke, where they specifically do not translate into pause statements and therefore do not change the semantics of such programs. Eventually, the same question arises in the conclusion in 8 where you state "The fundamental idea is to convert real-time wait constructs into logical waits (pause) that INTERACT WELL with the rest of the constructs in these languages [referring to Esterel]). In a similar vein, I'd appreciate a comparison with Multiclock Esterel (see e.g. Berry/Sentovich, CHARME '01). |  |
|  | - I would add "synchronous languages" to the keyword list | It is added as suggested |
|  | - In the Motivation 1.1 it is not entirely clear to me what the actual motivation is. You say "You would like to add a similar mechanism for in reactive languages" but not why. This point gets more and more clear later on, however it misses in the initial motivation. | We have extended this paragraph and explained that reactive languages lack in such features and why these mechanisms are needed in the real-time applications. |
|  | - The first time "statically" is mentioned is on page 3 in the contribution sub-section. I would recommend to make it clear earlier (in the introduction or even the abstract) that the introduced concepts transforms the real-time wait statements statically. | We have added this explanation in both abstract and at the end of introduction. |
|  | - page 6, in 3.3 4. and page 6, in 3.4: "any value of d": Which one is chosen exactly? Is always the lower bound chosen? So, as d is determined statically, d is fixed. Also you're saying it is trivial in 3.4 but it is not clear which d is chosen. And if the lower bound is chosen, where lies the difference between wait\_inbetween and wait\_atleast? | Our approach does not limit whether d should be fixed or can be changed during runtime, as long as the value d satisfies the real-time specified by wait\_inbetween or wait\_atleast constructs. Therefore, d does not necessarily to be the lower bound. |
|  | - page 12, in 5: In the description of the experimental results you state, that you allowed the compiler to relax the upper real-time bounds. Is there a particular reason you chose this setting? I ask because you stated earlier that this is the non-default case and that the compiler uses the periodic execution as default. | This option was chosen to show how our algorithm can accurately generate the results compared to the original delay specified by a designer. For example, we have shown in Figure 11 that how much the upper bound N should be relaxed (increased) in order to obtain a valid “d”. We have also discussed in Section 5.1 that this relaxation can be reduced, resulting close to the original delay specification, by shortening a critical path. |
|  | - Throughout the paper the wait commands and their parameters are printed with and without a space in between. Should be checked for consistency. | These are fixed |
|  | - page 1, keyword list is missing (placeholders are present) | Keywords are inserted |
|  | - page 2, in 1.1.1, par 3: First mentioning of Esterel: reference? | Reference added |
|  | - page 3, in 1.1.3: Contrary to the first two definitions of the wait statements, wait\_exact is not defined with an (M) (inconsistency) | Fixed |
|  | page 4, in 2.2.: "Unfortunately, execution..." Why is this unfortunately? Is DELTA not the unfortunate part? | Fixed |
|  | - page 5, in 3.2, Figure 5: Is 'while' not a kernel construct? Why can't one just write while(x<d) { pause; x=x+1; } | Only while(true){…} is **not** a kernel construct. SystemJ programmer can only break this loop using trap or abort statements. |
|  | - page 7, in 3.4/3.5: repetition: You stated two times, that inserting a pause breaks critical paths. | The second repetition is removed from text. |
|  | - page 8, in 3.5.1: The first two paragraphs describe the TP. Is this part of this work? If not, maybe you can add a cite here. |  |
|  | > - page 14, in 5.1: Here, you state again that inserting pause statements help with critical paths. Maybe you should simply reference your earlier remark. "As stated in 3.4 adding addition pause statements..." | The statement is added |
|  | - page 1, in 1: I would begin a new paragraph before "In this paper..." |  |
|  | - figures and/or captions of figures are sometimes not aligned properly (eg fig. 2a, caption fig. 1a, more...) | Fixed |
|  |  |  |

# Reviewer 3

|  |  |  |
| --- | --- | --- |
|  | **Comments** | **Suggestions** |
| 3.01 |  |  |