Reviewer(s)' Comments to Author:  
Reviewer: 1  
  
Comments to the Author  
This paper is an extension to a conference paper published at APLAS 2018, as noted in the paper and in the attached extension report.  
  
The original conference paper proposes a practical Hoare-style program logic for verifying SPARC assembly code, based on previous work formalizing the operational semantics of SPARC. The SPARC assembly language is widely used in embedded systems. It has three special features making verification of programs difficult: delayed control transfers, delayed writes, and rotation of register windows. An extension of separation logic is proposed to handle these complexities. The logic is applied to verify the main body of a context switch routine.  
  
The main addition compared with the conference version is a relational program logic for verifying refinement between high-level and low-level SPARC programs. Here, a high-level SPARC language is defined, mainly by abstracting away low level details, in particular enforcing common sequences of instructions, and forbidding arbitrary writes to special registers. Then, the refinement relation is defined and a logic for proving refinement is proposed.  
Both the Hoare logic and the relational logic for refinement are important progress toward being able to effectively verify assembly programs written in SPARC. However, there are still several aspects where the paper can be improved.

The link between C and Assembly is not defined (as noted in the introduction),

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Yes, we do not define the link between C and Assembly in our work. As we have explained in the introduction of our paper (in the first paragraph on the right side of page 3), the goal of this paper is to verify the correctness of the assembly code with respect to the abstract assembly primitives, we want to avoid the C-assembly interaction and decompose the OS verification tasks into two steps as shown in Fig.2 (on page 3). We focus on the step (2) shown in Fig.2 in this paper and remain the step (1) as a future work about verifying the correctness of the compilation.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

and the low-level assembly used in the refinement appears to be different from the one defined in Section 3 (as noted at the beginning of Section 4.2). This makes it unclear how the different parts can be linked together.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

We make some supplementary explanations for the low-level SPARCv8 program in Sec. 4.2 in this version. (1) We change the form of program transition in Sec. 3 to “(C, S, pc, npc) ::==> (C, S’, pc’, npc’)”, which is the same as the program transition shown in Sec. 4.2. The program transition in Sec. 3 in previous version is in the form of “C |- (S, pc, npc) |--> (S’, pc’, npc’)”. We hope that this modification will let people know that our low-level is a complete SPARC program as defined in Sec. 3; (2) We also claim the differences between the SPARCv8 program acting as a low-level in refinement verification shown in Sec.4.2 and the one defined in Sec. 3 on page 17 (colored in blue).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

In particular, there are very little changes to Section 5, making it difficult to understand how the refinement calculus in Section 4 is applied to the case of the context switch routine.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

We add the proof sketch of context switch routine, including the lemmas and final theorem, and use some figures to illustrate our verification work in this version. We also report what is exactly the additional proof effect in this verification work comparing with the one presented in our conference paper. The supplementary can be found on page 27-29 (colored in blue).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

There are also quite a few typos and grammatical mistakes in Section 4 (some of which are listed below), making the submission appear quite rushed.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Thanks, we have carefully checked the spelling and fixed the typos in our paper.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
Overall, I feel the amount of additional material is barely enough for an extension to a journal article. For the SETTA conference, I consider it as a borderline submission.  
  
Minor suggestions:  
- Page 9, line 5R: legel -> legal.  
- Page 13, line 56R: two part -> two parts.  
- Page 14, line 26L: saving in -> saved in?  
- Page 14, line 5R: make … preserves -> make … preserve.  
- Page 14, line 35R: should be carefully, for example -> should be done carefully. For example.  
- Page 15, line 22L: We -> we.  
- Page 15, line 27L: windows unused -> unused windows.  
- Page 15, line 6R: in simple instruction -> as simple instructions?  
- Page 15, line 22R: is a a pair -> is a pair.  
- Page 15, line 41R: a list of pair -> a list of pairs.  
- Fig. 16 caption: Seletcted -> Selected.  
- Page 16, line 44R: are very closed -> is very close.  
- Page 21, line 54L: specifid -> specified.  
- Page 21, line 55R: some knowledges -> some knowledge.  
- Page 22, line 8L: satisfies -> that satisfies (or satisfying).  
- Page 22, line 24R: assemly -> assembly.  
- Page 22, line 28R: and it -> and itself.  
  
  
Reviewer: 2  
  
Comments to the Author  
The paper describes a framework to verify the SparkV8 assembly code. The assembly code is used in some OS kernel implementation to access hardware and to improve efficiency. People have proposed an approach to verify OS kernel code, in which assembly code is replaced with logic formulae expressing their semantics. The author formalized the behaviors of the SparkV8 code in this paper, which complements the previous approach under a refinement framework. That is, people can first show the kernel code is correct with assembly code replaced with logic formulae describing its semantics, then show a refinement relation between the logic formula and the concrete semantics provided in this paper.  
  
In general, I think this is a nice work with significant practical impact and should be accepted for publication. One common issue for correctness proof based on theorem provers is the high-demand in human intervention.

As future work, maybe the author can consider to automate the reasoning for certain parts with automatic provers like Z3 or some integration with framework like Frama-C, to reduce the required human effort in the correctness proof.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Thanks, we agree with the suggested ideas and claim that we will try to use some automatic provers to automate the reasoning for certain parts of our verification work in the future in the conclusion of our paper in this version (colored in blue in right side of page 30).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
Typo:   
page 3, right column, Finally, we can get (the refinement relation). In this work....   there is some typo in the refinement relation  
some typo in reference [23] and [24] PeterW -> Peter

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Thanks, we have carefully checked the spelling and fixed the typos in our paper.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Reviewer: 3  
  
Comments to the Author  
Modular Verification of SPARCv8 Code  
  
The paper presents a verification approach for inline assembly code in C programs. It presents the overall verification setup to split verification of the assembly code from its verification in the context of the C program by introducing an intermediate Pseudo-SPARC language and a refinement proof calculus for it. The modularity of the approach is restricted to this separation between languages and focuses solely on the first step (refinement verification of SPARC code).  
  
The approach is demonstrated with a small examples of a context switch routine as it occurs in operating system implementations.  
  
The paper does a descent job in explaining the challenges that are unique to some of SPARC's features, how they are represented in the intermediate language, and handled by the proof calculus.

However, the description of the register rotation on page 2 is somewhat abstract, and becomes much clearer with Fig. 1. Consider a small illustration early in the paper, or perhaps refer to Fig. 1 from the introduction.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

We refer to Fig. 1 to make a supplementary explanation of window rotation in this version in the first paragraph on the right side of the page 2 (colored in blue).

(\* 我其实没太看明白 reviewer 的这条建议。Reviewer 说第二页关于 window rotation 的介绍太抽象了，可以考虑提前或结合 Fig. 1 做一些说明。但之前第二页关于 window rotation 就是结合 Fig. 1 介绍的，所以我这里的修改只是对第二页里结合 Fig. 1 介绍 window rotation 的内容做了补充。 \*)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Also, since the present paper is an extension of a prior ASPLOS conference paper, an overview listing the main extension points (with references to sections and lemmas in the paper) would help to better assess the contributions of the present paper and to guide readers to the relevant sections (for example, on p. 4 before the paper overview).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Thanks, in this version, we use a list to itemize our expansions on the right side of Page.4 (colored in blue) before the paper overview and point which section gives detailed introduction to this part of work.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The refinement setup on p. 3 is a centerpiece of the overall approach. Please identify the formal characterization $\forall \mathcal{C} \mathcal{C}[C\_{as}] \sqsubseteq \mathcal{C}[A]$ with an equation number and refer to it from the text (e.g., after "as the following form").

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

We identify the formal characterization $\forall \mathcal{C} \mathcal{C}[C\_{as}] \sqsubseteq \mathcal{C}[A]$ with an equation number and refer to it from the text (after “as the following form”) in this version.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Please also more carefully explain the refinement proof chain. Most importantly: the final result of [[ \mathcal{C}[\Omega] ]]^\mathsf{C} \sqsubseteq [[ C[C\_{as}] ]]^\mathsf{SPARCv8} seems not what we want to have (and it is unclear why it would follow from the refinement chain in Fig. 2). Please elaborate and/or fix.  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Thanks,

the [[ \mathcal{C}[\Omega] ]]^\mathsf{C} \sqsubseteq [[ C[C\_{as}] ]]^\mathsf{SPARCv8} is a typo, and we have fixed it as [[ C[C\_{as}] ]]^\mathsf{SPARCv8} \subseteq [[ \mathcal{C}[A] ]]^\mathsf{C}. (Note that in this version, we use “A \subseteq B” to represent “A refine B” and use “A \sqsubseteq B” to represent “A contextually refine B”).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The paper generally does explain syntax/semantics well in the text. Some small suggestions related to Fig. 3: SimpIns seems not described in the text. be f seems not described in the text.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

We supplement the descriptions of SimpIns and “be f ” in this version, which can be found on the right side of Page.5 (colored in blue). We also add the semantics (in Fig.7 on Page.9) and the inference rule (in Fig.11 on Page.13) of “be f ”.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Suggestions related to Fig. 6: please briefly mention the notation for lists and tuples in the text (::, .).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

We add the explanation of the meaning of **::** and **·** in this version (on the right side of page 7 colored in blue).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Fig. 9: what does M\_1 \bot M\_2 mean?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

We add the formal definition of “M\_1 \bot M\_2” in this version (in Fig.9 on page 11).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Fig. 11: please briefly mention Hoare-triple syntax in the text.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

We briefly mention syntax and the structure of our program logic in this version (in the second paragraph on the left side of page 12).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Fig. 12: difficult to read in black-and-white print, consider using patterns in addition to color.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Thanks. In this version, we use the light gray to mark the windows saving contexts of pervious procedures, the dark gray to mark the stack in memory saving the contexts of previous procedures and the east north lines to fill the invalid window. Please find the modifications in Fig.12 on page 14 and Fig.13 on page 15 (we use dots to fill the original invalid window in Fig.13).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Fig. 18: why is in line mathcal{p}\downarrow (R',D')-->(R,D)?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

We supplement the explanations of the meaning of “\matcal{p}\downarrow” and why there is (R’, D’) --> (R, D) in the last paragraph on the right side of page 19 (in color blue).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sect. 4 seems to be the extension over the ASPLOS conference paper. Please comment what exactly is new.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

At the beginning of Sec. 4, we declare what exactly is new of our expansion (colored blue in page 4) in this version.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Please also provide proof sketches for your lemmas and theorems.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

We supplement the proof sketch of our sound proof in this version. Please find them on page 24-26 (colored in blue).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
Sect. 5 introduces the case study. While the general verification setup is illustrated, it remains unclear how much manual effort that proof requires in Coq. Please elaborate, e.g., by presenting a proof sketch or referring to the Coq proof.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

We have to admit that this part of work is not mechanized in Coq. In this version, we claim this fact and supplement the proof sketch of context switch routine, which can be found on page 27-29 (colored in blue). The proof of context switch routine, which consists of around 250 lines of SPARCv8 code, presented in our conference paper is mechanized in Coq. We prove it by 6690 lines of Coq proof script. We claim what is exactly the additional proof effect of verification work in this work comparing with the one presented in our conference paper.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
Overall, the paper is written quite well and the intermediate SPARC-language is introduced in an approachable way, but the paper may benefit from improvements (proof sketches, better and more explicit highlighting of new contributions, better illustration of the verification effort when using the approach in Sect. 5). Some minor comments follow below (the list is long, but should all be easily fixable).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Thanks, we have carefully checked the spelling and fixed the typos in our paper.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Minor editorial comments  
------------------------  
1. p. 1: delayed write feature, please briefly comment why it is useful and that this basically forces nop instructions for portability (as later explained on p. 2)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

In the introduction of delayed write feature in page 1 in this version, we briefly comment the use of delayed write (execution of instruction wr, which is used to write special registers, takes 0~3 cycles to complete and causes delayed write) and mention that the number of delay instructions is implementation-dependent.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2. p. 2: let npc points to --> lets npc point to  
3. p. 2 (At line 6 [...] value of %i\_0): perhaps mention that %i\_0 now points to %o\_0 of line 1  
4. p. 3: mention that $A \sqsubseteq B$ means "A refines B".  
5. p. 3: primtives --> primitives  
6. p. 3: As (1) shown --> As (1) shows  
7. p. 3: call abstract assembly primitive in \Omega --> call abstract assembly primitives in \Omega  
8. p. 4: implementated --> implemented  
9. p. 4: rest of paper --> remainder of the paper  
10. p. 5: saved in code heap --> saved in the code heap  
11. p. 5: paramters --> parameters  
12. p. 5: control-transfer instructions like call and jmp. --> control-transfer instructions like call, jmp, retl, and be.  
13. p. 6: by decrements --> by decrementing  
14. p. 8: Fig. 7(a) caption: Transistion --> Transition  
15. Fig. 7 caption: please repeat that it is taken from [7]  
16. p. 9: legel --> legal  
17. p. 9: We define syntax --> We define the syntax  
18. p. 10: one delayed writes --> one delayed write

19. p. 11: lgvl is mentioned in the text but undefined

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

This is a typo, we use (\iota) to represent the list of values, which plays the role of auxiliary variables. We have modified the lgvl to .

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

20. p. 11: %l7 at the end --> %l7 in the beginning contains any value and at then end  
21. p. 12: contains two part --> contains two parts  
22. p. 14: are saving in register window --> are saved in register windows  
23. p. 14: code preserving such --> code preserves such  
24. p. 14: switch routine whose executions will store the contexts saved in register window into stack in memory cannot --> switch routine, whose executions will store the contexts saved in register windows into the stack in memory, cannot  
25. p. 14: make [...] preserves --> make [...] preserve  
26. p. 14: modifying them should be carefully --> should be modified carefully  
27. p. 15: abstraction, We --> abstraction, we  
28. p. 15: pseudo instruction --> pseudo instructions  
29. p. 15: rephrase to "restrict the save and restore instructions such that they can only be used in the specific form mentioned before".  
30. p. 15: states of high-level Pseudo-SPARCv8 program is --> states of a high-level  Pseudo-SPARCv8 program are  
31. p. 15: \Gamma following --> \Gamm as follows

32. p. 16: occur empty message/occur a message (occur seems the wrong verb)  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

We have modified “occur” to “generate”.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

33. p. 16: in primitive set --> in the primitive set

34. p. 16: consider renaming execi to exec. What does =\_H mean?  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

We rename execi to exec in this version. In the previous version, we use “ execi( i, S) =\_H S’ ” to represent that executing instruction i from state S will reach state S’. The “\_H” is a just notation in syntax to represent executing instruction in high-level. In this version, we use “exec(i, S, S’)” to represent that executing instruction i from state S will reach state S’.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

35. p. 16: block b size from 64 byte to w byte --> block b of size 64 byte to w byte  
36. p. 16: in convention --> by convention  
37. p. 16: are very closed to --> are very close to  
38. Fig. 16 caption: Seletcted --> Selected  
39. p. 17: is splitted --> is split  
40. p. 17: modify special register by wr instruction --> modify any special register through wr instructions

41. consider using different symbols for contextual refinement vs. event-trace refinement

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

In this version, we use “” (sqsubseteq) to represent the contextual refinement and “” (\subseteq) to represent event-trace refinement.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

42. p. 18: parametered --> parameterized  
43. p. 18: paramterized --> parameterized  
44. p. 19: properities --> properties  
45. p. 20: primtive --> primitive  
46. p. 20: knowledges --> knowledge  
47. p. 20: we can know --> we know  
48. p. 21: Then we can know --> Then we in turn know  
49. Def. 3: if C\_as(ps)=i then: there exist S',pc,npc' (pc should be pc')  
50. p. 22: exists [...] (S,\mathcal{S},A,w) satisfies --> exists [...] (S,\mathcal{S},A,w) that satisfies  
51. p. 22: we have the following holds --> then also the following holds  
52. p. 22: assemly --> assembly

53. p. 22: in code heap C\_{as} and it (what does "it" refer to?)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

We fix “between its low-level implementation in code heap in code heap C\_{as} and it” to “between its low-level implementation in code heap in code heap C\_{as} and itself”. The “it” in the previous version refers to “abstract assembly primitive”.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

54. p. 23: lies in the code manages --> lies in the how the code manages  
55. p. 23: use nst to present --> use nst to represent  
56. p. 24: becomes non-current-thread --> becomes a non-current thread  
57. p. 24: as specified Rdy --> as specified by Rdy  
58. p. 24: in final state --> in the final state  
59. p. 24: without meet much --> without much  
60. p. 24: of SPARC machine language --> of the SPARC machine language  
61. p. 24: rephrase to "Since neither Wang et al. nor we validate the formalization against actual hardware, this remains future work."  
62. p. 24: show case --> showcase  
63. p. 25: supports main features --> supports the main features  
64. p. 26: please double check spacing in author name Peter W. O'Hearn