1. Each accepted paper should be accompanied with a full registration (by any one of the co-authors). For example, if one person has two accepted papers, s/he needs to make two full registrations. If any paper does not accompany a full-registration by the registration deadline, the paper will be removed from the conference. You will receive a separate email with registration details (including deadline), and it will also be posted in the conference webpage.
http://embeddedandvlsidesignconference.org/

2. When you prepare the camera-ready version of your paper (up to 6 pages), please address all the reviewer concerns (except the comments that you do not agree). Reviews comments are at the end of this email.

3. Please make sure that you and your co-authors are aware of IEEE plagiarism policy when you submit your camera-ready version.
https://www.ieee.org/publications/rights/plagiarism/plagiarism.html

4. You need to prepare the camera-ready version using IEEE conference formatting guidelines (including PDFxpress check), and submit the IEEE copyright form by the camera-ready submission deadline. We are still waiting for IEEE to open the PDFxpress website for VLSI Design 2019. Once we hear from them, hopefully in next two weeks, you will receive an email, and it will also be posted in the conference webpage.

5. Please prepare a set of high-quality slides and plan to deliver an excellent presentation in the conference. Soon, we will send guidelines on how to prepare presentation slides for 20 minute presentation.

6. Finally, one of the co-authors should present the paper in the conference. In case of a no-show, the paper will be removed from the IEEE Xplore Digital Library.

The technical program will be posted soon in the conference webpage. The main conference will be during Jan 7-9, 2019 at New Delhi. The hotel has not been decided yet. You will soon receive information regarding the venue, discounted hotel rates as well as procedure to get a VISA support letter (if applicable).

Please check the conference webpage regularly for any updates. If you have any specific questions related to the technical program, please contact the Program Chairs atvlsid2019@gmail.com. For all other questions, please contact the General Chairs at vlsidelhiteam@gmail.com

Congratulations once again for the acceptance of your paper at VLSI Design 2019! We look forward to meeting you at VLSI Design 2019 in New Delhi from January 5-9, 2019.

Thanks and regards,
Prabhat Mishra and Jayadeva
Program Chair, VLSI Design 2019

```
======================================================================
VLSID 2019 Reviews for Submission #47
======================================================================
```

Title: Improving Performance of Path Based Equivalence Checker using Counter-example
Authors: Ramanuj Chouksey, Chandan Karfa and Purandar Bhaduri

```
======================================================================
            REVIEWER #1
======================================================================
```

Detailed Comments to Authors

When a path-based equivalence checker reports a pair of paths from two FSMDs to be non-equivalent, no counter-example starting from a reset state is provided. So the reported non-equivalence can be a false negative due to unreachability from a reset state. The paper proposes a simple solution to this issue by using BMC to find a counter-example from a reset state that passes through both paths and produces some differences in output or state variables. Unfortunately, the technical details are spare so that it is not clear why doing so is better than e.g. just using BMC for bounded equivalence checking (which should clearly be less scalable?). The main insight seems to be in Section III, which IMHO should be much more thoroughly and carefully discussed. Section V, Case 3.2 : also unclear why the 2 FSMDs producing the same outputs on a concrete counter-example can be considered not a non-equivalent case? Experimental results are also not quite convincing. The considered design!
 s appear to be rather small. The gain of integrating the proposed method into a path-based equivalence checker is not pronounced. In ME cases, other falsification approaches might be able to find a counter-example faster. The real added value of the proposed method should actually be in the cases where path-based equivalence checking can be further advanced (ideally until a proof is found).

==
REVIEWER #2
==

Detailed Comments to Authors

The paper aims to generate a counter trace in case of non-equivalence.

Pros
The problem is well defined and relevant.
The solution analysis is sound covering various cases of equivalence, non-equivalence, false-negative, and its limitation.
The results look promising on the given benchmark.

Cons
The paper does not discuss CBMC. Specifically, why don't we use always CBMC if it provides accurate results than PBEC?
It's not clear if PBEC reports false-positive since it claimed that verification not complete. If yes, how does the proposed solution respond?

==
REVIEWER #3
==

Detailed Comments to Authors

Path-based equivalence checker (PBEC) is used for verification of high-level synthesis tool. This paper shows that how to generate counter-trace using the internal information of verifier in the case of non-equivalence reported by the PBEC. It also presents how to use CBMC to find a counterexample. Moreover, it shows how counterexample can be used to improve the performance of PBEC.

Although the topic of this paper is interesting, the following parts need further improvement or explanations.

1: 1: There are some typos and grammar errors in this paper, e.g., “The EVP method also report ...”. Please scan the paper carefully and correct all of them.

2: The example shown in figure 3 needs indentation.

3: In the experiment section, i am confused with “We manually introduce few changes in benchmark tabulated in rows 3-6 of Table I”. Please give more explanations.

4: I think the size of benchmarks shown in table I is too small. Please use some larger ones. Moreover, please provide the information of their RTL counterparts. I also want to see some concrete examples of “ME” and ”NE” in the experiment section.

==
REVIEWER #4
==

Detailed Comments to Authors

The article discusses the use of the well-known C Program Analysis tool CBMC to generate counter-examples to decide cases of potential equivalence between programs. The authors focus on equivalence checks for programs translated using High-Level synthesis (HLS) tools, that is equivalence between the source program for an HLS tool and the translation resulting from using the HLS tool. Each program is translated into an equivalent finite state machine with data-paths (FSMD). The authors use the existing EVP methodology ([7] in reference list) to identify path pairs that are unconditionally equivalence, conditionally equivalent and non-equivalent. Since, results associated with non-equivalence may be false negatives (due to the general undecidability of the equivalence checking problem for programs), the authors propose using CBMC on path pairs reported as conditionally equivalent to further distinguish cases that are truly non-equivalent from those that are equivalent (still p!
 ossibly ending with a “maybe equivalent” result). The authors use standard examples to test their methodology and demonstrate the differences in results arising from presence and absence of their plugin when used with the EVP methodology.

The use of CBMC for generating counter-examples is well known. It has been intelligently used in conjunction with EVP for better ascertaining equivalence. Further-more use of a counterexample to prove non-equivalence is useful to understand issues with HLS tools. I find the flow that has been proposed interesting.

None-the-less, I find the presentation and language used poor, despite having an interesting methodology being presented.

1. The abstract is unclear. The first statement “Path-based equivalence checker (PBEC) is used for verification of high-level-synthesis tool” is grammatically incorrect and seems to be placing PBEC tools as used only for verifying translations produced by HLS tools. The abstract does not provide sufficient information to the reader to gauge what the article is proposing (for instance – the idea that there can be situations where a “maybe equivalent” or “conditionally equivalent” result needs further analysis is unclear).

2. The paper lacks a section on related work on using counter-example guided analysis for programs. I find this quite surprising given that the article proposes to use exactly this to bolster its claim of improving the information resulting from a PBEC analysis for equivalence.

3. The use of “source” and “transformed behaviors” does not read well and can create confusions. A better emphasis on the relation between these terms in the definition of an HLS tool would make for a clearer reading.

4. The language used in the introduction is full of grammatical and structural errors, thereby taking eclipsing the motivation for the article.

5. In the introduction, contributions (1) and (2) seem identical. In contribution (3), perhaps “performance” is not the correct term here. From the results, the proposed methodology (when used) takes orders of magnitude more time than simply using PBEC. This is understandable since the proposition is to provide more information for non-equivalence results, however making a “performance” claim is not appropriate.

6. In Section II, there needs to be deeper explanations with examples for “FSMD”, a path, cutpoints, path-covers using diagrams and examples. The text description is messy and creates confusion for the reader. In the last paragraph of the section, the authors talk about mismatched variable values and propagated vectors but don’t discuss this with an example. Since the entire methodology proposed is based on the results from using EVP and FSMDs, understanding these techniques is paramount to understanding counterexample generation.

7. Section III is confusing to read as well. It is presented as a summary of the methodology, explaining how the results from applying EVP are used for counter-example trace generation. The explanation is messy and confusing. Having a running example would have helped, along with better formalizations for paths and path-covers. Furthermore, the authors begin by explaining equivalence with regards to only one path \alpha. Equivalence, in my understanding would be between two paths \alpha and \beta. The authors introduce \beta towards the end of the section.

8. Figure 2 seems redundant. This is already contained in example 1.

9. Figure 3 contains a program that is un-indented and difficult to read. The authors may consider shrinking the text contained therein and better indenting the program. This is crucial to understanding the input presented to CBMC.

10. In Section V the various cases resulting from using CBMC could benefit from better presentation styles. An example: Cases 3.2 and 3.3 are marginally distinguishable, however Case 3.1 is contained inside the para for Case 3.

11. Section VI does not contribute much to the overall presentation. Section VI algorithmically described what is already discussed in earlier sections. I would have preferred better toy examples and explanations in earlier sections.

==
REVIEWER #5
==

Detailed Comments to Authors

The paper presents improvements in path-based equivalence checking, by analyzing the results of conditional equivalency, to prove the actual equivalency or non-equivalency, if possible. The experimental results confirm the effectiveness of the method.
It would be helpful to report the size of the models for BMC, and the “k” (number of times to unwind) for previously reported ME cases in Table I.
