Top of Form







Accept Reject

**150-KW626 A Verilog Precompiler for Interactive Optimization of IP Core Design**

The principal reason(s) for this decision is (are) as follows:  
  
Paper has to be heavily revised by taking into account suggestions provided by the reviewers. There are serious concerns about the practicality of your approach. Your claims are not supported by enough empirical evidence. Your paper is very hard to understand or read.  
  
[View Reviews](javascript:__doPostBack('ctl02',''))

|  |
| --- |
| **1=unacceptable; 2=poor; 3=marginal; 4=good; 5=excellent**  150-KW626 A Verilog Precompiler for Interactive Optimization of IP Core Design  Reviewer ID: 54518 Overall Recommendation: 1 - Reject How novel or worthwhile are the ideas in the papers? 1 How adequately is the existing literature referenced? 1 How clearly is the technical contribution described? 1 How well are the claims and conclusions supported? 1 How well are the advantages and limitations described? 1 How well is the paper organized? 1 How clear is the author's use of English? 1 How effectively does the author use tables, figures and pictures? 1  COMMENTS TO THE AUTHOR The paper proposes a set of macros extending the Verilog language aiming at the improvement of the designer's productivity. The authors start from the assumption that parametric code cannot be written in Verilog and so they propose a pre-processor able to manipulate the extended Verilog code by introducing tree new macros: fold, map and forloop. Actually, this can be similarly done by using the generate block statement available from Verilog 2001 (e.g., google: verilog generate if or generate for). Generate statements allow parametrization of the Verilog and if need may be used to perform design space exploration. The author should consider that almost all the RTL synthesis tools perform full or partial loop unrolling taking into account different cost metrics: area, performance, power, etc. So, most of the design exploration steps discussed by the authors are actually already performed by the existing commercial tools.   Some further specific comments:  The authors instead of use the term reconfigurable should speak of parametrization. Moreover, the pre-compiler is actually a pre-processor.  please check the following sentences: Abstract: "And for different optimization choices, the circuits \*vary\* in architectures besides in time and resource."  "so designers are liberated from putting much effort on \*concerning\* about module scheduling and wire con- nection."  "which work on \*single kind\* of algorithm,"  Introduction: "Some designers \*go as far\* as using Perl "  "For \*industry-strength\* design, "  "lack of static checks at compile time in most HDL".  VHDL is a strongly typed language. Verilog, is not a strongly typed language but almost all the existing compilers for synthesis or simulation do several check before doing anything else.  "it is not easy to reconfigure the functionality of an existing design". Parametrization is possible both in Verilog and in VHDL.  "such unrolling cannot be achieved in Verilog without substantial code change.". It is not true. Beside generate for even the for statement can be used to compactly describe repeating statements.  "systems that translate high level mathematical representa- tions (in the SPL language) of certain DSP transforms into hardware designs". What do you mean with "DSP transforms".  "Take another \*examplme\* from Verilog, when designing a calculator"    ============================================================  Reviewer ID: 40955 Overall Recommendation: 1 - Reject How novel or worthwhile are the ideas in the papers? 2 How adequately is the existing literature referenced? 3 How clearly is the technical contribution described? 1 How well are the claims and conclusions supported? 1 How well are the advantages and limitations described? 2 How well is the paper organized? 2 How clear is the author's use of English? 2 How effectively does the author use tables, figures and pictures? 4  COMMENTS TO THE AUTHOR This paper presents new syntax additions to Verilog and an accompanying compiler to generate hardware for logic described with this new syntax. The additional language elements enable the designer to describe parameterizable units.  The paper describes what the endproduct (the precompiler that is) is capable of doing and fall short in explaining in any detail how the precompiler has been constructed and how and why these specific three language extensions (map, fold, loop) have been chosen against anything else. Is there any intuition based on any systematic analysis or profiling of a good set of applications to claim that these operations were the most common? Most amenable to pursuing trade-offs? There is basically no systematic approach one can identify in the way the authors handled the problem. The authors emphasize the "interactive" nature of the precompiler, however, they provide no details on how the interaction between the tool and the designer is implemented. Is the designer explicitly notified of the total number of choices available for the parameters? Is the designer asked to iterate one by one? enter parameter ranges and values manually? What compiler infrastructure has been used to implement this tool??   The authors should have reported run time for their examples. In practice, should this compiler run through all parameter combinations for all cases? This could quickly become expensive to do in terms of runtime? Is there any intelligence built into the tool to steer the designer away from unfavorable combinations?   The paper is poorly written overall, with too many typos, grammar mistakes and bad style and use of informal phrases. It needs serious editing. The same reference is repeated twice [13] and [20].  ?   ============================================================  Reviewer ID: 16720 Overall Recommendation: 1 - Reject How novel or worthwhile are the ideas in the papers? 2 How adequately is the existing literature referenced? 4 How clearly is the technical contribution described? 2 How well are the claims and conclusions supported? 3 How well are the advantages and limitations described? 2 How well is the paper organized? 1 How clear is the author's use of English? 2 How effectively does the author use tables, figures and pictures? 2  COMMENTS TO THE AUTHOR 1. This work proposed some algorithm level syntax to enhance Verilog for IP architectural optimization. 2. Actually there are many similar and better enhancements in HLS tools today using pragmas. Therefore I have to say there is no much innovation in this work. 3. Please be careful in the English writing. There exist typos and incorrect use of terms. The biggest mistake is the use of "embed." SV+ should be "embedded in," but "embedded with," Verilog. 4. Do not use company names as tool names. For example it is "Design Compiler" but 'Synopsys' that does logic synthesis. Synopsys has more than 6000 tools, covering from system level to lithography.  ============================================================  Reviewer ID: 46991 Overall Recommendation: 1 - Reject How novel or worthwhile are the ideas in the papers? 1 How adequately is the existing literature referenced? 2 How clearly is the technical contribution described? 2 How well are the claims and conclusions supported? 1 How well are the advantages and limitations described? 2 How well is the paper organized? 3 How clear is the author's use of English? 1 How effectively does the author use tables, figures and pictures? 3  COMMENTS TO THE AUTHOR This work presents an intermediate language for abstracting the circuit independently of the target language, which in this case is Verilog. In this way, the same piece of code can be implemented with multiple trade-offs choices. Nevertheless, I do not see the difference between using C and a tool like Catapult-C, or even a Verilog/VHDL behavioral description with different optimization scripts for Synopsys, Cadence or whatever commercial tool. Moreover, the paper must improve its readability.  Here are my concerns:  1) Several Haskell-like structures are proposed for increasing the abstraction of the design. For example, the case of figure 1 corresponds with a multiplier. What is the difference between using A\*B or fold(sum,var,2^n,2^(n-1),...,2^1) ? Unless underlying optimizations among several operations are easier to apply in the fold case, I do not see the advantage.  2) In the experiments section, comparisons against a different tool or flow should be performed, in order to prove the advantages of the proposed solution. Once again, different trade-offs can be obtained with different scripts, this is not new.  3) The paper is poorly written. Several paragraphs needs rewriting and misprints being corrected. For instance, 3rd paragrapgh or 2nd paragrapth belonging to sections 1.1 and 2.1, respectively. Besides, the use of informal constructions as he/she (pp. 3) is completely discouraged.  ============================================================  Reviewer ID: 51990 Overall Recommendation: 2 - Possible Reject How novel or worthwhile are the ideas in the papers? 4 How adequately is the existing literature referenced? 3 How clearly is the technical contribution described? 5 How well are the claims and conclusions supported? 5 How well are the advantages and limitations described? 4 How well is the paper organized? 4 How clear is the author's use of English? 1 How effectively does the author use tables, figures and pictures? 4  COMMENTS TO THE AUTHOR This paper presents SV+ which consists of a functional language and a compiler?framework to generate Verilog. The most interesting part of this work is the ease with?which user can do design space exploration like sacrificing area to gain throughput and?vice-versa. The user can provide some knobs based on which the compiler produces different?configurations in Verilog. I will categorize this work as high level synthesis (HLS).?Several HLS languages and compiler frameworks have been ?proposed which exploit different?forms of parallelism present in an application. Some of the tools give you the ability todo some form of design space exploration (like extent of loop unrolling, fusion, etc).?SV+ directly competes against that. It relies on three programming constructs, namely fold,?map and forloop. The syntax that SV+ uses looks similar to that of Verilog which makes?it easier for people who know HDL to use it. However, It may not be exciting and convenient?enough for the wider software community. The paper is well organized and the experimental evaluation is acceptable.? - The biggest drawback of this paper is the terrible use of English. It is simply unacceptable.?Even the section title (4.1.1 Implementatin of AES) is misspelled. - I find use of the term "reconfigurable" ?misleading. The paper doesn't explicitly tell what it?means by that term. My understanding is that the term "reconfigurable" in this paper refers to??a design object for which multiple configurations (or implementations) can exist. I suggest??authors to either find a better term or clearly explain the meaning of it.? - The authors do not explain the nature of optimization choices for a design. I? assume it consist of area and throughput metric but i would like to see more? detail on this. - Also, I didn't understand how the tool generates N configurations from an abstractfunction. For example, in Section 2.1 there is an abstract function fold(sum,var,sn,. . .,s0) which can lead to multiple implementations based on design choice provided byuser. What I don't understand is how and without that knowledge I am not so sure how theframework is going to work.?   ============================================================  Reviewer ID: 28572 Overall Recommendation: 1 - Reject How novel or worthwhile are the ideas in the papers? 2 How adequately is the existing literature referenced? 3 How clearly is the technical contribution described? 2 How well are the claims and conclusions supported? 1 How well are the advantages and limitations described? 2 How well is the paper organized? 1 How clear is the author's use of English? 3 How effectively does the author use tables, figures and pictures? 3  COMMENTS TO THE AUTHOR In general, the authors fail to motivate the need for a Verilog precompiler. They mention that a drawback of other languages (e.g., Bluespec) is that they require the designer to learn a new language. And yet this seems to be the same thing for this work. The constructs they introduce seem useful in a limited and small number of situations. And finally, the paper is full of grammar and spelling errors.? How did you define algorithmic level specification? Because one could argue that there are many high level synthesis tools that do this. And you mention many of these tools in the related work section.  The fold, map and forloop constructs need better motivation. Why did you select these and how often do these occur in hardware designs? A lot of grammar errors. Needs a serious proofreading. it is hard to take an article seriously when the abstract along is full of errors. Examples include:1) "resource consumption" not "consuming"2) Poor phrasing "They can be used to embed with Verilog to describe"3) "The compiler generates Verilog RTL codes, depends on these choices"   ============================================================ |

[Customer Support](mailto:papers@dac.com)

Bottom of Form