DRILLS: Debugging RTL Intelligently with Localization from Long-Simulation

VIGHNESH IYER and DONGGYU KIM, UC Berkeley

With increasing RTL design complexity it is increasingly difficult to debug why a subset of tests fail in chip-level emulation. We propose the use of specification mining to

1 INTRODUCTION

Specification mining is a technique to extract LTL properties from a set of traces of signals. This technique can be applied for several purposes in the domain of digital hardware verification including:

(1) Developing a suite of assertions to be used for design regressions: once a design is mature, most of the interface boundary specifications are well defined and fine-grained assertions derived from specification mining can help catch regressions when design refinements or optimizations are being made.
(2) A starting point for formally specifying a design: once a design can pass random-stimulus based and directed unit tests, specification mining can be used to extract properties that have been consistently observed in the test waveforms. These properties can then be used as assertions to prove formally.
(3) Early anomaly detection and localization on long-running tests on mature RTL: if a specific test fails on an RTL design while many other tests pass, specification mining can reveal *where and when* a failing test begins to produce unusual behavior in the RTL, guiding the designer to the bug location.

In this report, we will focus on applying specification mining to address the last point above.

1.1 Motivation

RTL designs are increasing in complexity and are thus more prone to having subtle bugs that are not caught in regular verification flows. Typical techniques such as randomized-stimulus testing, directed testing, and fuzz testing have a difficult time catching bugs that require the RTL be put into a very specific state.

These subtle bugs are usually caught when performing chip emulation or FPGA prototyping when running realistic workloads on the RTL. Real workloads usually involve traces that are billion of cycles long, and are thus too slow to perform using RTL simulation which provides full design visibility. DESSERT[19] demonstrates a technique to capture full-visibility waveform traces from fast running FPGA simulation. Using DESSERT, an out-of-order RISC-V processor (BOOM[5]) is deterministically emulated on an FPGA with runtime assertion monitors while running the SPEC2017 benchmark suite. During the execution of several tests, synthesized assertions were violated which revealed there exists some subtle bugs in the core causing some benchmarks to fail.

Authors' address: Vighnesh Iyer, vighnesh.iyer@berkeley.edu; Donggyu Kim, dgkim@berkeley.edu, UC Berkeley.

While these assertions are useful for catching errors, they are very high-level and don't direct the designer to where a bug originated. As an example, the "`Pipeline has hung`" assertion (in BOOM) is generated with the following Chisel code:

```
// detect pipeline freezes and throw error
val idle_cycles = freechips.rocketchip.util.WideCounter(32)
when (rob.io.commit.valids.asUInt.orR ||
      csr.io.csr_stall ||
      io.rocc.busy ||
      reset.asBool) {
  idle_cycles := 0.U
}
assert (!(idle_cycles.value(13)), "Pipeline has hung.")
```

In words, this says, "If there is a good reason to stall the pipeline, reset `idle_cycles`, otherwise let it tick up to 13 before declaring something has gone wrong." This assertion does not give any insight as to what bad event happened, or when and where it happened. Since these assertions are thrown after billions of cycles it is possible that some μ-arch state was corrupted early in the simulation and only triggered this assertion much later during execution. DESSERT enables extracting waveforms for a variable number of cycles before the assertion triggers, but even with the waveform dumps in hand, the designer was unable to localize the bug. Our aim in developing this specification mining tool is to hunt out the locations of these trickly bugs in BOOM and fix them.

1.2 Hypothesis

Mature RTL designs (like BOOM) pass almost all tests run on them, including a full set of ISA tests, a boot of Linux, and real applications running on an OS. If a test fails on a mature design, we hypothesize that an *assumption* the designer made about the RTL was violated somewhere and at some time during the failing test execution, that was not violated on any successful test execution. These assumptions can include believing that a certain register cannot hold certain values or higher-level properties such as: "the memory system will respond to my request within 5 cycles".

We believe specification mining can be used to extract designer assumptions about the RTL design by mining fine-grained LTL properties on waveforms of successful test executions. These mined properties can be added to the RTL design as assertions and replaying the failing test should cause a violation of a mined property. These violations can be used to catch a faulty assumption *earlier and with greater locality* than the high-level assertions originally present in the design.

1.3 Problem Definition

Given:

- An RTL design driven with only one global clock
- A large set of VCD (value change dump) files produced when running a full suite of tests in an RTL simulator
- One or just a few failing tests in the suite characterized by a failed high-level assertion, hanging/global timeout, or a bad exit code

Produce:

- A set of mined LTL properties involving signals (combinational nets and registers) of the RTL design that aren't violated on any passing test
- A method of ranking the mined LTL properties
- A program that can check a VCD against the mined properties to find any violations for that execution trace

2 APPROACH

Our plan is to first create the products above with a small RTL design. We use `riscv-mini`[18], a simple 3-stage in-order RISC-V processor as the RTL we will use to test the specification mining engine. `riscv-mini` has real instruction and data caches and implements the RV32UI ISA subset which is capable of running the entire `riscv-tests`[38] test suite. The test suite contains a comprehensive set of ISA tests and benchmarks which will be used to generate the VCD files for mining.

2.1 Prior Work

Specification mining has been applied to both software and hardware verification.

2.2 High-Level Flow

The long-term plan is to leverage the spec mining engine and the prior work of DESSERT to provide an FPGA-acclererated debugging platform. We plan to take waveforms of successful test executions from RTL simulation and FPGA emulation of BOOM and feed the spec mining engine to infer LTL properties. The buggy RTL is instrumented by taking the LTL properties and converting them to property monitor FSMs which are stiched into the original design. The assertions are synthesized for FPGA emulation using the same methodology described in DESSERT, and the failing tests are executed. It is our hope that the failing tests will trigger failures in the mined properties before the failure becomes visible to the user (via a high-level assertion violation, hanging/timout, or a bad exit code).

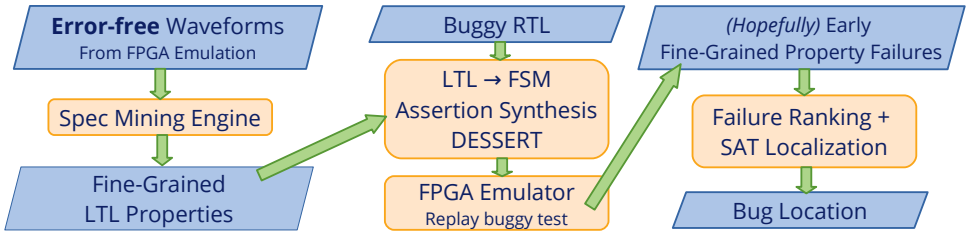

Fig. 1. The proposed tool flow to use specification mining and FPGA-accelerated simulation to pinpoint bug locations in for an RTL design .

The mined properties which were violated can be ranked based on the time of their violation and we can use the SAT localization techniques mentioned above to further pinpoint the bug location.

3 MODEL AND ALGORITHMS

In this section, we will specify the formal model for LTL specification mining, how we adapted LTL formulas for RTL, and the algorithms used in the spec mining engine.

3.1 Hardware Idioms in LTL

They can be used to describe many common idioms present in RTL. A few examples:

- There should eventually be a response (`resp`) after a request (`req`)
  $\mathbf{G}\,(\texttt{req} \rightarrow \mathbf{XF}\,\texttt{resp})$
- There should be a response in 2 cycles after a request
  $\mathbf{G}\,(\texttt{req} \rightarrow \mathbf{XX}\,\texttt{resp})$
- The ready/valid interface should keep `valid` high once it has been asserted until `ready` goes high
  $\mathbf{G}\,(\texttt{valid} \rightarrow \mathbf{X}\,(\texttt{valid}\,\mathbf{U}\,\texttt{ready}))$
- After a ready/valid transaction, the slave should be `ready` again within 2 cycles
  $\mathbf{G}\,((\texttt{valid} \wedge \texttt{ready}) \rightarrow (\mathbf{X}\,\texttt{ready} \vee \mathbf{XX}\,\texttt{ready}))$

3.1.1 LTL Templates. The formulas above can be templated by replacing the concrete signals (such as `ready`, `resp`, etc.) with variable placeholders. We consider 4 LTL templates for spec mining derived from Li's prior work[25]:

- Alternating: $a\,\mathbf{A}\,b$
- Until: $\mathbf{G}\,(a \rightarrow \mathbf{X}\,(a\,\mathbf{U}\,b))$
- Next: $\mathbf{G}\,(a \rightarrow \mathbf{X}\,b)$
- Eventual: $\mathbf{G}\,(a \rightarrow \mathbf{XF}\,b)$

where a and b are some boolean expressions derived from the signals in the RTL.

3.2 Adapting LTL Templates for RTL

RTL simulations produce a set of finite-length traces of bitvectors. We assume that these traces are sampled on the rising edge of a given clock signal; from now on we assume all traces contain the values of signals at discrete clock cycles. In contrast, LTL properties are defined over infinite-length traces of atomic propositions. We convert bitvector traces to traces of *delta events* which are treated as atomic propositions, and we employ notions of *falsifiability* and *support* to mine LTL properties on finite length traces.

Let a signal trace τ_i be a tuple of length T which contains the value of the signal i (a bitvector) at every discrete timestep of an RTL simulation. Let there be N signals in the RTL design: $i \in [0, N)$. We can convert τ_i to its delta trace $\tau_{\Delta i}$ as such:

$$\tau_{\Delta i}(t) = (\tau_i(t-1) \neq \tau_i(t)) \quad \forall t \in [1, T)$$

Note that $\tau_{\Delta i}$ is a tuple of length $T-1$ and is a trace of atomic propositions. Simply put, we convert a bitvector signal trace to a boolean trace which is 1 whenever the signal changes value, and 0 otherwise. We now consider the templates in section 3.1.1 in the context where $a, b = \tau_{\Delta i}, \tau_{\Delta j}, i \neq j$, for some i, j.

Here is a concrete example where k, q are bitvector signals in an RTL design where k is 1 bit wide and q is 10 bits wide:

$$\begin{aligned}
\tau_k &= (0, 1, 1, 1, 0) \\
\tau_q &= (0, 0, 200, 200, 300) \\
\tau_{\Delta k} &= (1, 0, 0, 1) \\
\tau_{\Delta q} &= (0, 1, 0, 1)
\end{aligned}$$

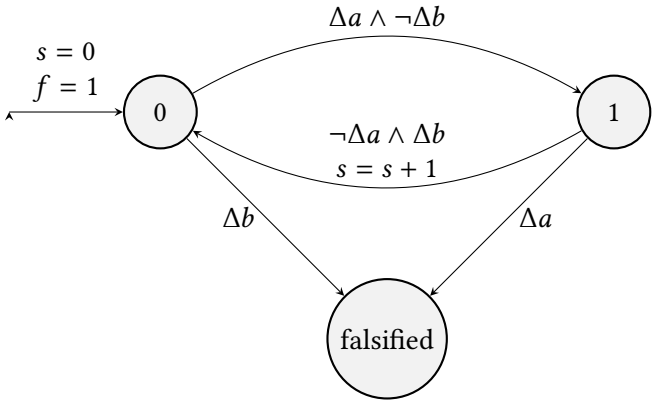

Fig. 2. Automaton that mines the alternating LTL pattern

Note that even though k is a boolean signal and can already be treated as an atomic proposition, we only mine LTL properties on its *delta trace* and not the signal itself.

In RTL designs, it is usually the case that the density of transitions for a given signal is fairly low. Signal traces can be stored in a compressed format where we only record the timestep in which the signal value changes. Indeed, this is conveniently how traces are stored in a VCD (value change dump) file which is the input to the spec miner. We can extract sparsely represented delta traces from a VCD file as a tuple of timesteps where the signal transitions.

$$\tau_{\Delta k, compressed} = (0, 3)$$
$$\tau_{\Delta q, compressed} = (1, 3)$$

We want to consider the templates in section 3.1.1, so we can zip 2 compressed delta traces together that indicates which of the signals transitioned.

$$\tau_{\Delta k, \Delta q} = ((\Delta k, \neg \Delta q), (\neg \Delta k, \Delta q), (\Delta k, \Delta q))$$

This means that initially, k transitioned and q did not on a given timestep, then on a future timestep q transitioned and k did not, and on another future timestep both k and q transitioned. Note that in this data structure, it is impossible to have a tuple where both k and q *did not* transition. This data structure is perfect to construct automatons that mine LTL properties.

3.3 Mining with Automata

We construct FSMs for each of the LTL property templates that take zipped compressed delta traces as an input.

3.3.1 Falsifiability and Support. A given LTL property template is *falsifiable* over a zipped delta trace if its mining automaton reaches a state from which it can move to the *falsified* state in one step. We loosely define the *support* for an LTL property over a trace as the number of times the pattern "repeats". It will be made explicit in the FSMs for each LTL template as to when a pattern "repetition" occurs; we have yet to formalize this notion. We will abbreviate falsifiable as f and support as s.

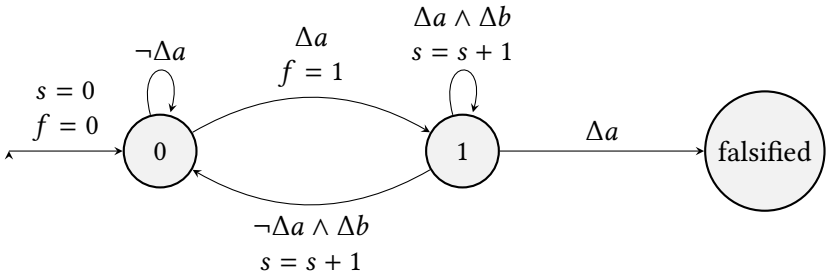

Fig. 3. Automaton that mines the until LTL pattern

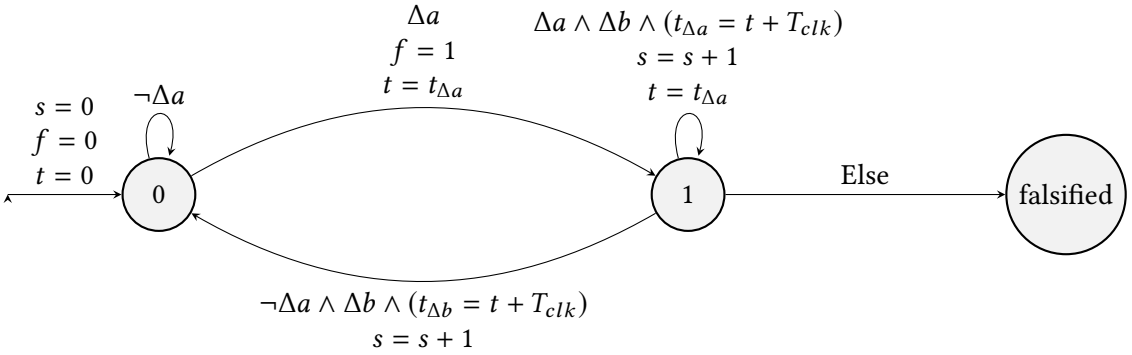

Fig. 4. Automaton that mines the next LTL pattern

3.3.2 Alternating. The automaton in figure 2 mines the alternating pattern and advances support when a Δa then Δb sequence occurs. This pattern is violated if Δa and Δb are both true on the same timestep.

3.3.3 Until. The automaton in figure 3 mines the until pattern. If Δa is seen 2 times in a row without a Δb then this pattern is falsified. Note that if Δa never occurs, this pattern can't be falsifiable on that particular delta trace.

3.3.4 Next. Some hacking is needed with the next automaton in figure 4 to constrain the pattern to only be valid when Δb happens after 1 clock cycle after Δa. The variables $t_{\Delta a}$ and $t_{\Delta b}$ represent the time at which the a and b delta events occured on the transitions in which they are used. Let T_{clk} be the clock period, and t is a state variable to hold the transition time of a.

3.3.5 Eventual. Note that the eventual pattern is never falsifiable on a finite-length trace as seen in figure 5. In addition to simply keeping track of the *support* we also maintain a set of times it took to get to Δb from Δa. This allows us to empirically find cycle limits for the eventual pattern so it can reasonably be applied to finite-length traces.

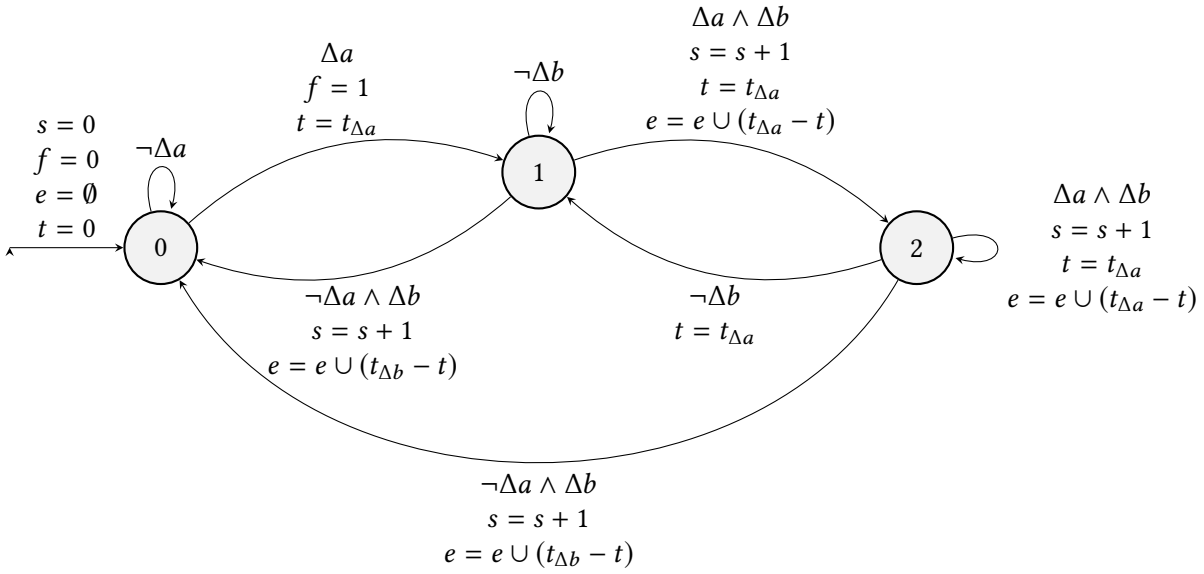

Fig. 5. Automaton that mines the eventual LTL pattern

Journals use one of three template styles. All but three ACM journals use the `acmsmall` template style:

- `acmsmall`: The default journal template style.
- `acmlarge`: Used by JOCCH and TAP.
- `acmtog`: Used by TOG.

The majority of conference proceedings documentation will use the `acmconf` template style.

- `acmconf`: The default proceedings template style.
- `sigchi`: Used for SIGCHI conference articles.
- `sigchi-a`: Used for SIGCHI "Extended Abstract" articles.
- `sigplan`: Used for SIGPLAN conference articles.

5.4 Template Parameters

In addition to specifying the *template style* to be used in formatting your work, there are a number of *template parameters* which modify some part of the applied template style. A complete list of these parameters can be found in the *LaTeX User's Guide.*

Frequently-used parameters, or combinations of parameters, include:

- `anonymous,review`: Suitable for a "double-blind" conference submission. Anonymizes the work and includes line numbers. Use with the `\acmSubmissionID` command to print the submission's unique ID on each page of the work.
- `authorversion`: Produces a version of the work suitable for posting by the author.
- `screen`: Produces colored hyperlinks.

This document uses the following string as the first command in the source file: `\documentclass[sigconf]{acmart}`.

6 MODIFICATIONS

Modifying the template — including but not limited to: adjusting margins, typeface sizes, line spacing, paragraph and list definitions, and the use of the `\vspace` command to manually adjust the vertical spacing between elements of your work — is not allowed.

Your document will be returned to you for revision if modifications are discovered.

7 TYPEFACES

The "`acmart`" document class requires the use of the "Libertine" typeface family. Your TEX installation should include this set of packages. Please do not substitute other typefaces. The "`lmodern`" and "`ltimes`" packages should not be used, as they will override the built-in typeface families.

8 TITLE INFORMATION

The title of your work should use capital letters appropriately - https://capitalizemytitle.com/ has useful rules for capitalization. Use the `title` command to define the title of your work. If your work has a subtitle, define it with the `subtitle` command. Do not insert line breaks in your title.

If your title is lengthy, you must define a short version to be used in the page headers, to prevent overlapping text. The `title` command has a "short title" parameter:

```
\title[short title]{full title}
```

9 AUTHORS AND AFFILIATIONS

Each author must be defined separately for accurate metadata identification. Multiple authors may share one affiliation. Authors' names should not be abbreviated; use full first names wherever possible. Include authors' e-mail addresses whenever possible.

Grouping authors' names or e-mail addresses, or providing an "e-mail alias," as shown below, is not acceptable:

```
\author{Brooke Aster, David Mehldau}
\email{dave,judy,steve@university.edu}
\email{firstname.lastname@phillips.org}
```

The `authornote` and `authornotemark` commands allow a note to apply to multiple authors — for example, if the first two authors of an article contributed equally to the work.

If your author list is lengthy, you must define a shortened version of the list of authors to be used in the page headers, to prevent overlapping text. The following command should be placed just after the last `\author{}` definition:

```
\renewcommand{\shortauthors}{McCartney, et al.}
```

Omitting this command will force the use of a concatenated list of all of the authors' names, which may result in overlapping text in the page headers.

The article template's documentation, available at https://www.acm.org/publications/proceedings-template, has a complete explanation of these commands and tips for their effective use.

10 RIGHTS INFORMATION

Authors of any work published by ACM will need to complete a rights form. Depending on the kind of work, and the rights management choice made by the author, this may be copyright transfer, permission, license, or an OA (open access) agreement.

Regardless of the rights management choice, the author will receive a copy of the completed rights form once it has been submitted. This form contains LaTeX commands that must be copied into the source document. When the document source is compiled, these commands and their parameters add formatted text to several areas of the final document:

- the "ACM Reference Format" text on the first page.
- the "rights management" text on the first page.
- the conference information in the page header(s).

Rights information is unique to the work; if you are preparing several works for an event, make sure to use the correct set of commands with each of the works.

11 CCS CONCEPTS AND USER-DEFINED KEYWORDS

Two elements of the "acmart" document class provide powerful taxonomic tools for you to help readers find your work in an online search.

The ACM Computing Classification System — https://www.acm.org/publications/class-2012 — is a set of classifiers and concepts that describe the computing discipline. Authors can select entries from this classification system, via https://dl.acm.org/ccs/ccs.cfm, and generate the commands to be included in the LaTeX source.

User-defined keywords are a comma-separated list of words and phrases of the authors' choosing, providing a more flexible way of describing the research being presented.

CCS concepts and user-defined keywords are required for all short- and full-length articles, and optional for two-page abstracts.

12 SECTIONING COMMANDS

Your work should use standard LaTeX sectioning commands: `section`, `subsection`, `subsubsection`, and `paragraph`. They should be numbered; do not remove the numbering from the commands.

Simulating a sectioning command by setting the first word or words of a paragraph in boldface or italicized text is **not allowed.**

Table 1. Frequency of Special Characters

Non-English or Math	Frequency	Comments
Ø	1 in 1,000	For Swedish names
π	1 in 5	Common in math
$	4 in 5	Used in business
Ψ_1^2	1 in 40,000	Unexplained usage

Table 2. Some Typical Commands

Command	A Number	Comments
`\author`	100	Author
`\table`	300	For tables
`\table*`	400	For wider tables

13 TABLES

The "`acmart`" document class includes the "`booktabs`" package — https://ctan.org/pkg/booktabs — for preparing high-quality tables.

Table captions are placed *above* the table.

Because tables cannot be split across pages, the best placement for them is typically the top of the page nearest their initial cite. To ensure this proper "floating" placement of tables, use the environment **table** to enclose the table's contents and the table caption. The contents of the table itself must go in the **tabular** environment, to be aligned properly in rows and columns, with the desired horizontal and vertical rules. Again, detailed instructions on **tabular** material are found in the *LaTeX User's Guide*.

Immediately following this sentence is the point at which Table 1 is included in the input file; compare the placement of the table here with the table in the printed output of this document.

To set a wider table, which takes up the whole width of the page's live area, use the environment **table*** to enclose the table's contents and the table caption. As with a single-column table, this wide table will "float" to a location deemed more desirable. Immediately following this sentence is the point at which Table 2 is included in the input file; again, it is instructive to compare the placement of the table here with the table in the printed output of this document.

14 MATH EQUATIONS

You may want to display math equations in three distinct styles: inline, numbered or non-numbered display. Each of the three are discussed in the next sections.

14.1 Inline (In-text) Equations

A formula that appears in the running text is called an inline or in-text formula. It is produced by the **math** environment, which can be invoked with the usual `\begin . . . \end` construction or with the short form `$ . . . $`. You can use any of the symbols and structures, from α to ω, available in LaTeX [23];

this section will simply show a few examples of in-text equations in context. Notice how this equation: $\lim_{n\to\infty} x = 0$, set here in in-line math style, looks slightly different when set in display style. (See next section).

14.2 Display Equations

A numbered display equation—one set off by vertical space from the text and centered horizontally—is produced by the **equation** environment. An unnumbered display equation is produced by the **displaymath** environment.

Again, in either environment, you can use any of the symbols and structures available in LaTeX; this section will just give a couple of examples of display equations in context. First, consider the equation, shown as an inline equation above:

$$\lim_{n\to\infty} x = 0 \tag{1}$$

Notice how it is formatted somewhat differently in the **displaymath** environment. Now, we'll enter an unnumbered equation:

$$\sum_{i=0}^{\infty} x + 1$$

and follow it with another numbered equation:

$$\sum_{i=0}^{\infty} x_i = \int_0^{\pi+2} f \tag{2}$$

just to demonstrate LaTeX's able handling of numbering.

15 FIGURES

The "`figure`" environment should be used for figures. One or more images can be placed within a figure. If your figure contains third-party material, you must clearly identify it as such, as shown in the example below.

Your figures should contain a caption which describes the figure to the reader. Figure captions go below the figure. Your figures should **also** include a description suitable for screen readers, to assist the visually-challenged to better understand your work.

Figure captions are placed *below* the figure.

15.1 The "Teaser Figure"

A "teaser figure" is an image, or set of images in one figure, that are placed after all author and affiliation information, and before the body of the article, spanning the page. If you wish to have such a figure in your article, place the command immediately before the `\maketitle` command:

```
\begin{teaserfigure}
  \includegraphics[width=\textwidth]{sampleteaser}
  \caption{figure caption}
  \Description{figure description}
\end{teaserfigure}
```

Fig. 6. 1907 Franklin Model D roadster. Photograph by Harris & Ewing, Inc. [Public domain], via Wikimedia Commons. (https://goo.gl/VLCRBB).

16 CITATIONS AND BIBLIOGRAPHIES

The use of BibTeX for the preparation and formatting of one's references is strongly recommended. Authors' names should be complete — use full first names ("Donald E. Knuth") not initials ("D. E. Knuth") — and the salient identifying features of a reference should be included: title, year, volume, number, pages, article DOI, etc.

The bibliography is included in your source document with these two commands, placed just before the `\end{document}` command:

```
\bibliographystyle{ACM-Reference-Format}
\bibliography{bibfile}
```

where "`bibfile`" is the name, without the "`.bib`" suffix, of the BibTeX file.

Citations and references are numbered by default. A small number of ACM publications have citations and references formatted in the "author year" style; for these exceptions, please include this command in the **preamble** (before "`\begin{document}`") of your LaTeX source:

```
\citestyle{acmauthoryear}
```

Some examples. A paginated journal article [2], an enumerated journal article [8], a reference to an entire issue [7], a monograph (whole book) [22], a monograph/whole book in a series (see 2a in spec. document) [14], a divisible-book such as an anthology or compilation [10] followed by the same example, however we only output the series if the volume number is given [11] (so Editor00a's series should NOT be present since it has no vol. no.), a chapter in a divisible book [34], a chapter in a divisible book in a series [9], a multi-volume work as book [21], an article in a proceedings (of a conference, symposium, workshop for example) (paginated proceedings article) [3], a proceedings article with all possible elements [33], an example of an enumerated proceedings article [12], an informally published work [13], a doctoral dissertation [6], a master's thesis: [4], an online document / world wide web resource [1, 28, 35], a video game (Case 1) [27] and (Case 2) [26] and [24] and (Case 3) a patent [32], work accepted for publication [29], 'YYYYb'-test for prolific author [30] and [31]. Other cites might contain 'duplicate' DOI and URLs (some SIAM articles) [20]. Boris / Barbara Beeton: multi-volume works as books [16] and [15]. A couple of citations with DOIs: [17, 20]. Online citations: [35–37].

17 ACKNOWLEDGMENTS

Identification of funding sources and other support, and thanks to individuals and groups that assisted in the research and the preparation of the work should be included in an acknowledgment section, which is placed just before the reference section in your document.

This section has a special environment:

```
\begin{acks}
...
\end{acks}
```

so that the information contained therein can be more easily collected during the article metadata extraction phase, and to ensure consistency in the spelling of the section heading.

Authors should not prepare this section as a numbered or unnumbered `\section`; please use the "`acks`" environment.

18 APPENDICES

If your work needs an appendix, add it before the "`\end{document}`" command at the conclusion of your source document.

Start the appendix with the "`appendix`" command:

```
\appendix
```

and note that in the appendix, sections are lettered, not numbered. This document has two appendices, demonstrating the section and subsection identification method.

19 SIGCHI EXTENDED ABSTRACTS

The "`sigchi-a`" template style (available only in LaTeX and not in Word) produces a landscape-orientation formatted article, with a wide left margin. Three environments are available for use with the "`sigchi-a`" template style, and produce formatted output in the margin:

- `sidebar`: Place formatted text in the margin.
- `marginfigure`: Place a figure in the margin.
- `margintable`: Place a table in the margin.

ACKNOWLEDGMENTS

To Robert, for the bagels and explaining CMYK and color spaces.

REFERENCES

[1] Rafal Ablamowicz and Bertfried Fauser. 2007. CLIFFORD: a Maple 11 Package for Clifford Algebra Computations, version 11. Retrieved February 28, 2008 from http://math.tntech.edu/rafal/cliff11/index.html

[2] Patricia S. Abril and Robert Plant. 2007. The patent holder's dilemma: Buy, sell, or troll? *Commun. ACM* 50, 1 (Jan. 2007), 36–44. https://doi.org/10.1145/1188913.1188915

[3] Sten Andler. 1979. Predicate Path expressions. In *Proceedings of the 6th. ACM SIGACT-SIGPLAN symposium on Principles of Programming Languages (POPL '79)*. ACM Press, New York, NY, 226–236. https://doi.org/10.1145/567752.567774

[4] David A. Anisi. 2003. *Optimal Motion Control of a Ground Vehicle.* Master's thesis. Royal Institute of Technology (KTH), Stockholm, Sweden.

[5] Christopher Celio, David A. Patterson, and Krste Asanovic. 2015. *The Berkeley Out-of-Order Machine (BOOM): An Industry-Competitive, Synthesizable, Parameterized RISC-V Processor.* Technical Report. https://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-167.html

[6] Kenneth L. Clarkson. 1985. *Algorithms for Closest-Point Problems (Computational Geometry).* Ph.D. Dissertation. Stanford University, Palo Alto, CA. UMI Order Number: AAT 8506171.

[7] Jacques Cohen (Ed.). 1996. Special issue: Digital Libraries. *Commun. ACM* 39, 11 (Nov. 1996).

[8] Sarah Cohen, Werner Nutt, and Yehoshua Sagic. 2007. Deciding equivalances among conjunctive aggregate queries. *J. ACM* 54, 2, Article 5 (April 2007), 50 pages. https://doi.org/10.1145/1219092.1219093

[9] Bruce P. Douglass, David Harel, and Mark B. Trakhtenbrot. 1998. Statecarts in use: structured analysis and object-orientation. In *Lectures on Embedded Systems*, Grzegorz Rozenberg and Frits W. Vaandrager (Eds.). Lecture Notes in Computer Science, Vol. 1494. Springer-Verlag, London, 368–394. https://doi.org/10.1007/3-540-65193-4_29

[10] Ian Editor (Ed.). 2007. *The title of book one* (1st. ed.). The name of the series one, Vol. 9. University of Chicago Press, Chicago. https://doi.org/10.1007/3-540-09237-4

[11] Ian Editor (Ed.). 2008. *The title of book two* (2nd. ed.). University of Chicago Press, Chicago, Chapter 100. https://doi.org/10.1007/3-540-09237-4

[12] Matthew Van Gundy, Davide Balzarotti, and Giovanni Vigna. 2007. Catch me, if you can: Evading network signatures with web-based polymorphic worms. In *Proceedings of the first USENIX workshop on Offensive Technologies (WOOT '07)*. USENIX Association, Berkley, CA, Article 7, 9 pages.

[13] David Harel. 1978. *LOGICS of Programs: AXIOMATICS and DESCRIPTIVE POWER.* MIT Research Lab Technical Report TR-200. Massachusetts Institute of Technology, Cambridge, MA.

[14] David Harel. 1979. *First-Order Dynamic Logic.* Lecture Notes in Computer Science, Vol. 68. Springer-Verlag, New York, NY. https://doi.org/10.1007/3-540-09237-4

[15] Lars Hörmander. 1985. *The analysis of linear partial differential operators. III.* Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], Vol. 275. Springer-Verlag, Berlin, Germany. viii+525 pages. Pseudodifferential operators.

[16] Lars Hörmander. 1985. *The analysis of linear partial differential operators. IV.* Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], Vol. 275. Springer-Verlag, Berlin, Germany. vii+352

pages. Fourier integral operators.
[17] IEEE 2004. IEEE TCSC Executive Committee. In *Proceedings of the IEEE International Conference on Web Services (ICWS '04)*. IEEE Computer Society, Washington, DC, USA, 21–22. https://doi.org/10.1109/ICWS.2004.64
[18] Donggyu Kim. 2019. Simple RISC-V 3-stage Pipeline in Chisel. https://github.com/ucb-bar/riscv-mini
[19] Donggyu Kim, Christopher Celio, Sagar Karandikar, David Biancolin, Jonathan Bachrach, and Krste Asanovic. 2018. DESSERT: Debugging RTL Effectively with State Snapshotting for Error Replays across Trillions of Cycles. In *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE. https://doi.org/10.1109/fpl.2018.00021
[20] Markus Kirschmer and John Voight. 2010. Algorithmic Enumeration of Ideal Classes for Quaternion Orders. *SIAM J. Comput.* 39, 5 (Jan. 2010), 1714–1747. https://doi.org/10.1137/080734467
[21] Donald E. Knuth. 1997. *The Art of Computer Programming, Vol. 1: Fundamental Algorithms (3rd. ed.).* Addison Wesley Longman Publishing Co., Inc.
[22] David Kosiur. 2001. *Understanding Policy-Based Networking* (2nd. ed.). Wiley, New York, NY.
[23] Leslie Lamport. 1986. *LaTeX: A Document Preparation System.* Addison-Wesley, Reading, MA.
[24] Newton Lee. 2005. Interview with Bill Kinder: January 13, 2005. Video. *Comput. Entertain.* 3, 1, Article 4 (Jan.-March 2005). https://doi.org/10.1145/1057270.1057278
[25] Wenchao Li. 2014. *Specification Mining: New Formalisms, Algorithms and Applications.* Technical Report. http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-20.html
[26] Dave Novak. 2003. Solder man. Video. In *ACM SIGGRAPH 2003 Video Review on Animation theater Program: Part I - Vol. 145 (July 27–27, 2003)*. ACM Press, New York, NY, 4. https://doi.org/99.9999/woot07-S422
[27] Barack Obama. 2008. A more perfect union. Video. Retrieved March 21, 2008 from http://video.google.com/videoplay?docid=6528042696351994555
[28] Poker-Edge.Com. 2006. Stats and Analysis. Retrieved June 7, 2006 from http://www.poker-edge.com/stats.php
[29] Bernard Rous. 2008. The Enabling of Digital Libraries. *Digital Libraries* 12, 3, Article 5 (July 2008). To appear.
[30] Mehdi Saeedi, Morteza Saheb Zamani, and Mehdi Sedighi. 2010. A library-based synthesis methodology for reversible logic. *Microelectron. J.* 41, 4 (April 2010), 185–194.
[31] Mehdi Saeedi, Morteza Saheb Zamani, Mehdi Sedighi, and Zahra Sasanian. 2010. Synthesis of Reversible Circuit Using Cycle-Based Approach. *J. Emerg. Technol. Comput. Syst.* 6, 4 (Dec. 2010).
[32] Joseph Scientist. 2009. The fountain of youth. Patent No. 12345, Filed July 1st., 2008, Issued Aug. 9th., 2009.
[33] Stan W. Smith. 2010. An experiment in bibliographic mark-up: Parsing metadata for XML export. In *Proceedings of the 3rd. annual workshop on Librarians and Computers (LAC '10)*, Reginald N. Smythe and Alexander Noble (Eds.), Vol. 3. Paparazzi Press, Milan Italy, 422–431. https://doi.org/99.9999/woot07-S422
[34] Asad Z. Spector. 1990. Achieving application requirements. In *Distributed Systems* (2nd. ed.), Sape Mullender (Ed.). ACM Press, New York, NY, 19–33. https://doi.org/10.1145/90417.90738
[35] Harry Thornburg. 2001. Introduction to Bayesian Statistics. Retrieved March 2, 2005 from http://ccrma.stanford.edu/~jos/bayes/bayes.html
[36] TUG 2017. Institutional members of the TeX Users Group. Retrieved May 27, 2017 from http://wwtug.org/instmem.html
[37] Boris Veytsman. [n.d.]. acmart—Class for typesetting publications of ACM. Retrieved May 27, 2017 from http://www.ctan.org/pkg/acmart
[38] Andrew Waterman. 2019. Unit Tests for RISC-V Processors. https://github.com/riscv/riscv-tests

A RESEARCH METHODS

A.1 Part One

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi malesuada, quam in pulvinar varius, metus nunc fermentum urna, id sollicitudin purus odio sit amet enim. Aliquam ullamcorper eu ipsum vel mollis. Curabitur quis dictum nisl. Phasellus vel semper risus, et lacinia dolor. Integer ultricies commodo sem nec semper.

A.2 Part Two

Etiam commodo feugiat nisl pulvinar pellentesque. Etiam auctor sodales ligula, non varius nibh pulvinar semper. Suspendisse nec lectus non ipsum convallis congue hendrerit vitae sapien. Donec at laoreet eros. Vivamus non purus placerat, scelerisque diam eu, cursus ante. Etiam aliquam tortor auctor efficitur mattis.

B ONLINE RESOURCES

Nam id fermentum dui. Suspendisse sagittis tortor a nulla mollis, in pulvinar ex pretium. Sed interdum orci quis metus euismod, et sagittis enim maximus. Vestibulum gravida massa ut felis suscipit congue. Quisque mattis elit a risus ultrices commodo venenatis eget dui. Etiam sagittis eleifend elementum.

Nam interdum magna at lectus dignissim, ac dignissim lorem rhoncus. Maecenas eu arcu ac neque placerat aliquam. Nunc pulvinar massa et mattis lacinia.