RISC-V International Documentation Template

Version 0.01, 04/2021: Pre-release version

Table of Contents

Preamble

This is the preamble, which can contain contributor information. For this book example, I am here mentioning that I have copied information directly from several Web sites that are hosted by asciidoctor.org and devoted to providing Asciidoc/Asciidoctor documentation. Graphics used are either explicitly available for free, are property of RISC-V International, or were created using Wavedrom.

Preface

The following is a preface for this document and should be replaced with a preface for your document.

This document demonstrates the use of Asciidoc for RISC-V specifications, with the goal of capturing information that will result in effective and efficient collaboration throughout the community.

Asciidoc is the most feature-rich of the popular lightweight markup languages based on Markdown. Most of the markup that you will need is simple, and much is similar to what you use for git-flavored Markdown.

It's helpful to think of Asciidoc as Markdown grown up. People in tech often have impulses to re-invent markdown with a brand new lightweight markup language of their own. As appealing as that idea can be, it is inherently flawed. Publishing, like music, can have simple forms, but when fully featured is quite complex. Everyone who has attempted to build upon Markdown to create a simple and feature-rich publishing solution faces the same reality.

RISC-V specifications require the use of Asciidoc markup and the Asciidoctor toolchain with advanced publishing features that are provided by several add-ons. The templates in this repo are here to allow you to jump in with a hands-on approach and build a pdf using the example files.

Because asciidoc is gaining in popularity, there are opportunities contributors to their specification while it is still being developed. You might want to view what Dan Allen and Sarah White, who support Asciidoctor along with the growing open source community, are currently doing. Feel free to find out about the working group, the specification under development, the various plugins, and ways in which Asciidoc is in use.

Copyright and licensure

Copyright and license information can go here.

Chapter 1. A few basics

Asciidoc is fully documented, and its documentation is actively maintained. This document contains some information on asciidoc markup to get you started.

For details and additional options:

- Asciidoc/asciidoctor writers' guide: asciidoctor.org/docs/asciidoc-writers-guide/
- Asciidoc quick reference: asciidoctor.org/docs/asciidoc-syntax-quick-reference/
- Asciidoctor user manual: asciidoctor.org/docs/user-manual/

In addition, you have the option of asking questions in the asciidoctor discussion list:

As is true of any complex process, asciidoc/asciidoctor has some quirks. Please be certain to make use of these templates because they provide you with files that will result in fully featured pdf output.

Best practice is to test the pdf build frequently to ensure that you have not accidentally introduced something that breaks the build.

PDFs require the use of Ruby 2.7.2.

1.1. Headers

In asciidoc you cannot jump directly from a Head 1 to a Head 3 or 4. Your headers must appear in numerical sequence from Head 1 to Head 2, and onward. If you skip over a header in the sequence, asciidoctor throws an error.

```
= Title head (book or report title)

[colophon]
= Colophon head (in frontmatter, used for preface)

[[chapter_title]]
== Head 1 (chapter)

=== Head 2 (section)

==== Head 3 (subsection)

===== Head 4 (sub-subsection)

[appendix]
== Appendix title

[index]
Index
```

Settings in the header file trigger auto-generation of Appendix prefixes and of the Index (among other things).

1.2. Code Blocks

Asciidoc/asciidoctor supports code blocks with syntax highlighting for many languages. You can use either periods or dashes to indicate code blocks, and use macros to indicate that the block contains code in the specified language, as in the following example:

```
[source,python]
....
mono-spaced code block
add a1,a2,a3; # do an ADD
....
```

renders as follows:

```
mono-spaced code block
add a1,a2,a3; # do an ADD
```

1.3. Hyperlinks and cross references

Asciidoctor recognizes hyperlinks to Web pages and shortens them for readability. Cross references (links within a document) are supported by macros.

1.4. Stem content

The `:stem: latexmath` setting works for HTML output from asciidoc, however, it doesn't yet work for asciidoctor-pdf output.

As a workaround, you will need to create image files rendered equations and import the images for pdfs.

```
[stem]
++++
sqrt(4) = 2
++++
```

```
sqrt(4) = 2
```

It looks like there is not much need for actual equations in the specifications. Rather, there is a set of specific unicode characters that are used:

Asciidoctor PDF currently only supports decimal character references. See github.com/asciidoctor/asciidoctor-pdf/issues/486

Hexadecimal unicode looks like it has problems in the pdf. This is gnarley.

Updates ti asciidoctor-pdf: github.com/asciidoctor/asciidoctor-pdf

1.4.1. Asciidoctor Mathematical

Asciidoctor Mathematical is an extension for Asciidoctor that provides alternate processing of STEM blocks and inline macros. After the document has been parsed, the extension locates each asciimath, latexmath, and stem block and inline macro, converts the expression to an image, and replaces the expression with an image. It uses Mathematical to render the LaTeX notation as an image. If the expression is AsciiMath, it first uses AsciiMath gem to convert to LaTeX. Conversion then proceeds as normal.

Asciidoctor Mathematical is a Ruby gem that uses native extensions. It has a few system prerequisites which limit installation to Linux and macOS. Please refer to the installation section in the Asciidoctor Mathematical README to learn how to install it.

Once Asciidoctor Mathematical is installed, you can enable it when invoking Asciidoctor PDF using the -r flag:

```
$ asciidoctor-pdf -r asciidoctor-mathematical sample.adoc
```

To get started, first create a Gemfile in the root of your project. In that file, declare the gem source, the asciidoctor-pdf gem, and the prawn-table gem (plus any other development gems you want to use).

Gemfile

source 'https://rubygems.org'

```
gem 'asciidoctor-pdf'
gem 'prawn-table', github: 'prawnpdf/prawn-table'
```

You can then install the gems into your project using the bundle command:

```
$ bundle config set --local path .bundle/gems && bundle
```

Since you're using Bundler to manage the gems, you'll need to prefix all commands with bundle exec. For example:

```
$ bundle exec asciidoctor-pdf -v
```

Thus, to any command present in the following sections, be sure to include this prefix.

Chapter 2. Table, symbols, and graphics

Asciidoc makes standard tables easy and also supports the creation of complex tables.

2.1. Some table examples

AsciiDoc tables can also be created directly from CSV data. Just set the format block attribute to CSV and insert CSV data inside the block delimiters directly:

```
[%header,format=csv]
|===
Artist,Track,Genre
Baauer,Harlem Shake,Hip Hop
The Lumineers,Ho Hey,Folk Rock
|===
```

The above renders as follows:

Artist	Track	Genre
Baauer	Harlem Shake	Hip Hop
The Lumineers	Ho Hey	Folk Rock

Here is an additional example of what can be done with tables:

```
[cols="e,m,^,>s",width="25%"]
|============================
|1 >s|2 |3 |4
^|5 2.2+^.^|6 .3+<.>m|7
^|8
|9 2+>|10
|============================
```

Which renders as follows:

1	**2**	3	**4**
5	6		
8			
9		10	7

Following is code for a numbered encoding table with link target.

Annotations have been added to the code to illustrate their use.

```
[#proposed-16bit-encodings-1]  ①
.proposed 16-bit encodings-1    ②
[width="100%",options=header]
|===
|15 |14 |13 |12 |11 |10 |9 |8 |7 |6 |5 |4 |3 |2 |1 |0 |instruction
3+|100|1|0|0|0 2+|field|0 |0 2+|00 | field 2+|00|mnemonic1
3+|100|1|0|0 3+|field|bit|1 3+|field 2+|00|mnemonic2
3+|110|1|0|0 3+|field|1 |0 3+|field 2+|00|mnemonic3
17+|This row spans the whole table
3+|100|1|1|1 8+|    field        2+| 00   | mnemonic4
|===
```

1. Link target.
2. Numbered table title.

Table 1. proposed 16-bit encodings-1

15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**instruction**
100			1	0	0	0	field		0	0	00		field	00		mnemonic1
100			1	0	0	field			bit	1	field			00		mnemonic2
110			1	0	0	field			1	0	field			00		mnemonic3
This row spans the whole table																
100			1	1	1	field								00		mnemonic4

2.2. Unicode symbols

For pdf, some unicode symbols are buggy. There are some workarounds. The asciidcotor-pdf toolchain supports most unicode up to and including four digits, and not characters that require five or more digits. I noticed the need for a mathematical *w* and because its encoding uses an integer that is in the 5+ digit category it doesn't work.

Here are a few unicode examples from en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references:

As an example, ◆ is encoded as follows:

```
&#9830;
```

Table 2. Useful unicode for specifications

sym	num	name
◆	9830	name
"	0034	name
w	0077	w
∴	8756	therefore
♯	9839	sharp
ш	1096	shcy
ϖ	982	piv varpi
ω	969	omega
℘	8472	weierp wp
∑	8721	sum
∞	8734	infin
∫	8747	integral
≠	8800	not equal to
≤	8804	le
≥	8805	ge
≈	8776	numerical approximation
D	68	mathematical D?
⇒	8658	rightwards double arrow
X	88	Latin Capital x
χ	967	Greek x
×	215	times
☑	9745	boxed checkmark
r	114	latin small letter r

Unfortunately a better checkmark is not available for asciidoctor-pdf, because anything above four digits doesn't work. The fact that encodings for moe than four digits for HTML encoding doesn't work is connected to prawn, which is used for generating the fully functional asciidoctor-pdf, and not to asciidoctor-pdf itself. Until the newer asciidoctor build that uses a different toolchain becomes fully featured, we must use a workaround.

It is possible to map fonts for better substitutes to numbers for which you don't need to make use of the existing unicode mapping should the need become a priority. For example, if a perfect mathematical *w* is important, we can implement the workaround until the newer toolchain for the pdf build is fully featured.

Table 3. Unicode identified as not working

sym	num	name
☒	9084	angzarr not working
☒	8921	ggg not working
☒	8617	hookleftarrow not working
☒	9083	not checkmark not working

2.3. Graphics

While asciidoc can render graphics in all popular formats, by far the highest quality graphics rendering is from .svg format.

WaveDrom sequence diagrams are essential to the RISC-V specifications. We are in the process of phasing in an automated process for incorporating WaveDrom diagrams into the professional quality pdf output so please stay tuned.

Asciidocdoctor-pdf enables automation of diagrams from scripts, including WaveDrom.

Even as we are using WaveDrom to simplify the creation of accurate svgs for register diagrams, the graphical elements—those for the various diagrams—add complexity to the build.

2.3.1. Automated diagramming

The asciidoctor-diagram extension supports numerous diagram types including WaveDrom diagramming (for sequence and waveform diagrams).

The requirements for building WaveDrom diagrams are specified in the docs temaplates readme.

The following json-formatted script, when added within an asciidoc block with the macro indicators `[wavedrom, svg]`, will embed the diagram output into the pdf:

```
{reg:[
    { bits:  7, name: 0x3b, attr: ['OP-32'] },
    { bits:  5, name: 'rd' },
    { bits:  3, name: 0x0, attr: ['ADD.UW'] },
    { bits:  5, name: 'rs1' },
    { bits:  5, name: 'rs2' },
    { bits:  7, name: 0x04, attr: ['ADD.UW'] },
]}
```

For the above to build into a diagram, with a figure title, you need to add the macro information and a figure title above the code block:

```
.Figure title
[wavedrom, svg]
```

Prior bug is fixed and requirements for local build added to the README. Once the required node and ruby extensions are installed, the diagrams build from asciidoc blocks

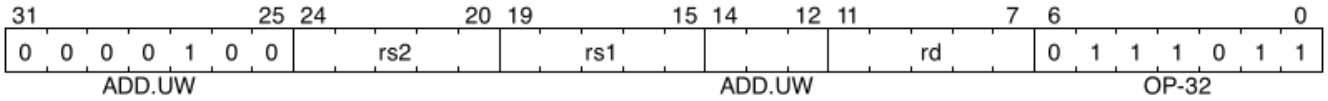


Figure 1. Figure title

You have the option of referencing a graphics file directly:

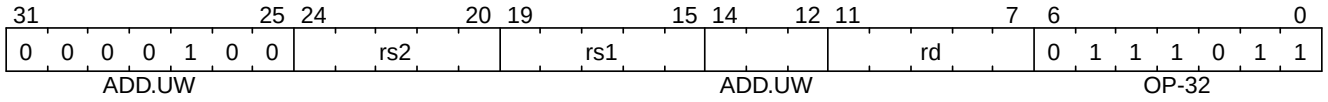


Figure 2. This example is from an svg generated prior to the build

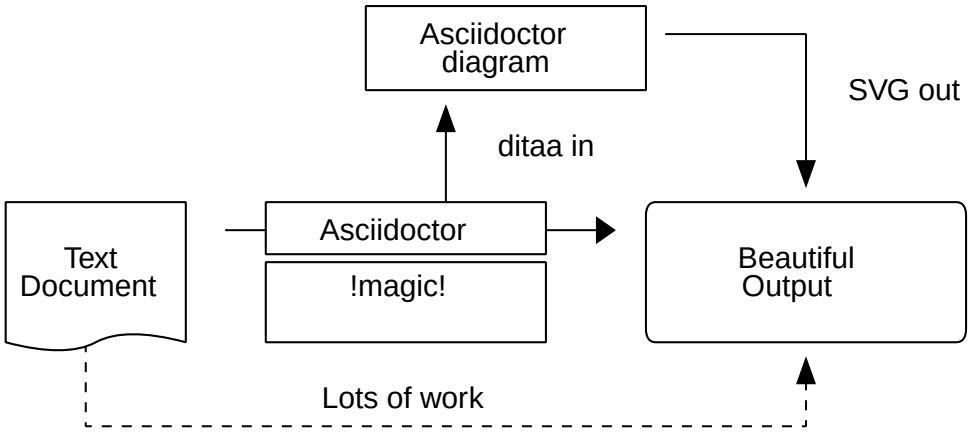
2.3.2. Additional diagram type examples

Following is source for simple ditaa diagram:

```
[ditaa,target="image-example",svg]
....

                  +-------------+
                  | Asciidoctor |-------+
                  |   diagram   |       |
                  +-------------+       | SVG out
                      ^                 |
                      | ditaa in        |
                      |                 v
 +--------+   +--------+----+    /---------------\
 |        | --+ Asciidoctor +--> |               |
 |  Text  |   +-------------+    |   Beautiful   |
 |Document|   |   !magic!   |    |    Output     |
 |     {d}|   |             |    |               |
 +---+----+   +-------------+    \---------------/
     :                                   ^
     |          Lots of work             |
     +-----------------------------------+
....
```

Which renders to:

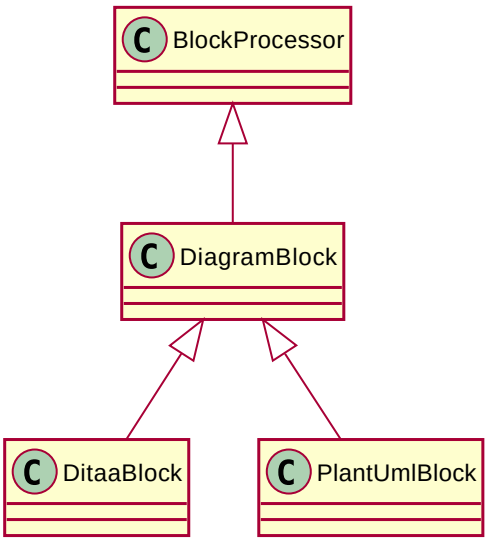


Following is source for a plantuml diagram:

```
[plantuml, diagram-classes, svg]
....
class BlockProcessor
class DiagramBlock
class DitaaBlock
class PlantUmlBlock

BlockProcessor <|-- DiagramBlock
DiagramBlock <|-- DitaaBlock
DiagramBlock <|-- PlantUmlBlock
....
```

Which renders to:



Chapter 3. Blocks, notes and markers

RISC-V specifications are notable for their extended commentaries that explain the thinking behind various important aspects of the technologies.

In most cases, contributors should make use of admonition blocks for their commentaries.

3.1. Blocks

Asciidoctor allows for many types of blocks, as documented at docs.asciidoctor.org/asciidoc/latest/blocks/.

3.1.1. Sidebars

Sidebars provide for a form of commentary.

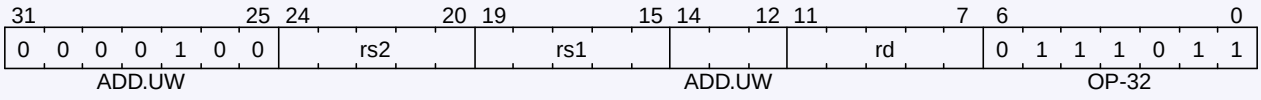
```
****
This is content in a sidebar block.

image:example-3.svg[]

This is more content in the sidebar block.
****
```

This renders as follows:

This is content in a sidebar block.



This is more content in the sidebar block.

You can add a title, along with any kind of content. Best practice for many of the "commentaries" in the LaTeX source that elucidate the decisionmaking process is to convert to this format with the `TIP` icon that illustrates a conversation or discussion, as in the following example:

```
.Optional Title
****
Sidebars are used to visually separate auxiliary bits of content
that supplement the main text.

TIP: They can contain any type of content, including admonitions like this, and
code examples like the following.

.Source code block in a sidebar
[source,js]
```

const { expect, expectCalledWith, heredoc } = require('../test/test-utils')

```
****
```

The above renders as:

Optional Title

Sidebars are used to visually separate auxiliary bits of content that supplement the main text.

They can contain any type of content, including admonitions like this, and code examples like the following.

Listing 1. Source code block in a sidebar

```
const { expect, expectCalledWith, heredoc } = require('../test/test-utils')
```

3.1.2. Admonition blocks

Five kinds of standard admonition blocks are available in asciidoc and these can be mapped to either default or custom icons.

```
[NOTE]
====
This is an example of an admonition block.

Unlike an admonition paragraph, it may contain any AsciiDoc content.
The style can be any one of the admonition labels:

* NOTE
* TIP
* WARNING
* CAUTION
* IMPORTANT
====
```

This renders as:

This is an example of an admonition block.

Unlike an admonition paragraph, it may contain any AsciiDoc content. The style can be any one of the admonition labels:

- NOTE
- TIP
- WARNING
- CAUTION
- IMPORTANT

For a single paragraph admonition, simply use a double colon:

```
NOTE: Note content.
```

which renders as:

Note content.

Alternate octicons:

- alert-24
- comment-discussion-24
- flame-24
- info-24
- pencil-24
- question-24
- sheild-24
- squirrel-24
- zap-24

Another example of an admonition block:

```
[IMPORTANT]
.Feeding the Werewolves
====
While werewolves are hardy community members, keep in mind the following dietary
concerns:

. They are allergic to cinnamon.
. More than two glasses of orange juice in 24 hours makes them howl in harmony
with alarms and sirens.
. Celery makes them sad.
====
```

Rendered:

Feeding the Werewolves

While werewolves are hardy community members, keep in mind the following dietary concerns:

1. They are allergic to cinnamon.
2. More than two glasses of orange juice in 24 hours makes them howl in harmony with alarms and sirens.
3. Celery makes them sad.

github.com/asciidoctor/asciidoctor-pdf/blob/master/docs/theming-guide.adoc#key-prefix-admonition-icon

The default admonition icons don't look right for RISC-V specification, and alternate icons and colors have been set in risc-v_spec-pdf.yml. and will be considered.

Current icons, edited to tone down color:

note

tip

warning

caution

important

Table 4. Customized colors for icons

Icon	default	customized
NOTE	19407c	6489b3
TIP	111111	5g27ag
WARNING	bf6900	9c4d4b
CAUTION	bf3400	c99a2c
IMPORTANT	bf0000	b58f5b

3.1.3. Code blocks

AsciiDoc enables code blocks that support syntax highlighting.

For example, preceding a block with a macro `[source,json]` enables `json` syntax highlighting:

```
{
    "weather": {
        "city":        "Zurich",
        "temperature":       25,
    }
}
```

3.2. Footnotes

Asciidoc has a limitation in that footnotes appear at the end of each chapter. Asciidoctor does not support footnotes appearing at the bottom of each page.

You can add footnotes to your presentation using the footnote macro. If you plan to reference a footnote more than once, use the footnote macro with a target that you identify in the brackets.

```
Initiate the hail-and-rainbow protocol at one of three levels:

- doublefootnote:[The double hail-and-rainbow level makes my toes tingle.]
- tertiary
- apocalyptic

A bold statement!footnote:disclaimer[Opinions are my own.]

Another outrageous statement.footnote:disclaimer[]
```

Renders as:

The hail-and-rainbow protocol can be initiated at three levels:

- double [1]
- tertiary
- apocalyptic

A bold statement! [2]

Another outrageous statement.[2]

3.3. Index markers

There are two types of index terms in AsciiDoc:

A flow index term. appears in the flow of text (a visible term) and in the index. This type of index term can only be used to define a primary entry:

```
indexterm2:[<primary>] or ((<primary>))
```

A concealed index term. a group of index terms that appear only in the index. This type of index term can be used to define a primary entry as well as optional secondary and tertiary entries:

```
indexterm:[<primary>, <secondary>, <tertiary>]
```

--or--

```
(((<primary>, <secondary>, <tertiary>)))
```

```
The Lady of the Lake, her arm clad in the purest shimmering samite,
held aloft Excalibur from the bosom of the water,
signifying by divine providence that I, ((Arthur)), ①
was to carry Excalibur (((Sword, Broadsword, Excalibur))). ②
That is why I am your king. Shut up! Will you shut up?!
Burn her anyway! I'm not a witch.
Look, my liege! We found them.

indexterm2:[Lancelot] was one of the Knights of the Round Table. ③
indexterm:[knight, Knight of the Round Table, Lancelot] ④
```

① The double parenthesis form adds a primary index term and includes the term in the generated output.

② The triple parenthesis form allows for an optional second and third index term and does not include the terms in the generated output (a concealed index term).

③ The inline macro `indexterm2\:[primary]` is equivalent to the double parenthesis form.

④ The inline macro `indexterm:\[primary, secondary, tertiary]` is equivalent to the triple parenthesis form.

If you're defining a concealed index term (the indexterm macro), and one of the terms contains a comma, you must surround that segment in double quotes so the comma is treated as content. For example:

```
I, King Arthur.
indexterm:[knight, "Arthur, King"]
```

I, King Arthur.

--or--

```
I, King Arthur.
(((knight, "Arthur, King")))
```

I, King Arthur.

3.4. Bibliography and references

You can add bibliographic entires to the last appendix that you use in a book document.

Text with markup that will generate links"

```
_The Pragmatic Programmer_ <<pp>> should be required reading for all developers.
To learn all about design patterns, refer to the book by the "`Gang of Four`"
<<gof>>.
```

Links from within text to bibliographic entries:

```
[bibliography]
== References

* [[[pp]]] Andy Hunt & Dave Thomas. The Pragmatic Programmer:
From Journeyman to Master. Addison-Wesley. 1999.
* [[[gof,gang]]] Erich Gamma, Richard Helm, Ralph Johnson & John Vlissides.
Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.
1994.
```

Text that links to bibliogpraphy:

The Pragmatic Programmer [pp] should be required reading for all developers. To learn all about design patterns, refer to the book by the “Gang of Four” [gang].

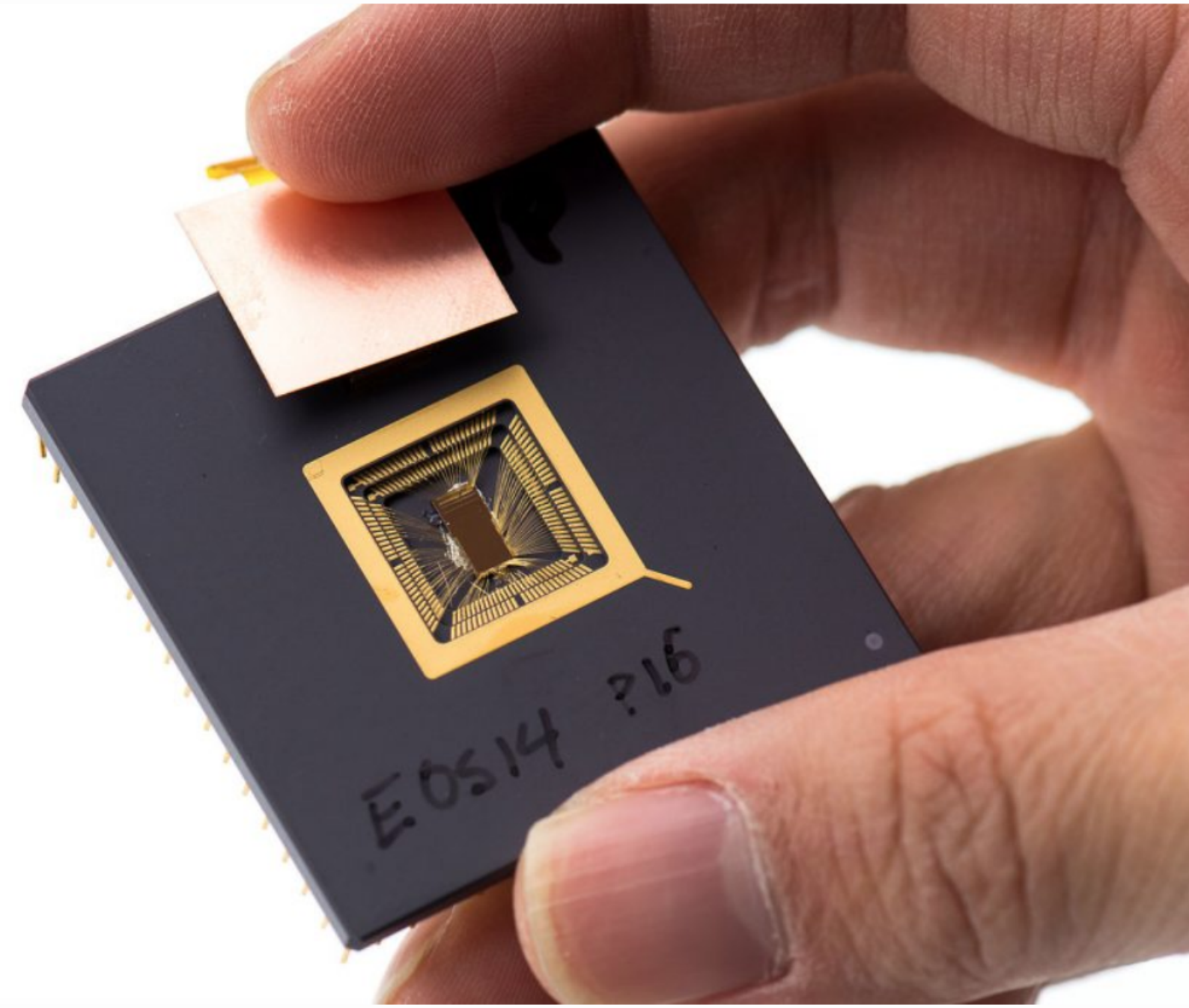
[1] The double hail-and-rainbow level makes my toes tingle.
[2] Opinions are my own.

Index

K

Bibliography

- [pp] Andy Hunt & Dave Thomas. The Pragmatic Programmer: From Journeyman to Master. Addison-Wesley. 1999.
- [gang] Erich Gamma, Richard Helm, Ralph Johnson & John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. 1994.


EOS14 ?16