RISC-V International Documentation Template

Version 0.01, 04/2021: Pre-release version

Table of Contents

Preamble

This is the preamble, which can contain copyright 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

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 througout 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 it 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/asciidoctor advanced publishing features. These templates are here to allow you to jump in with a hands-on approach and build the example pdf.

Because asciidoc is gaining in popularity, there are opportunities contributors to their specification while it is still being developeed. You might want to view what Dan Allen, who supports asciidoc/asciidoctor, is currently doing.

Asciidoc is gaining wider adoption. 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

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

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

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

1.3. Hyperlinks and cross references

blah blah

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. Tables 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

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. 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

sym	num	name
☒	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.

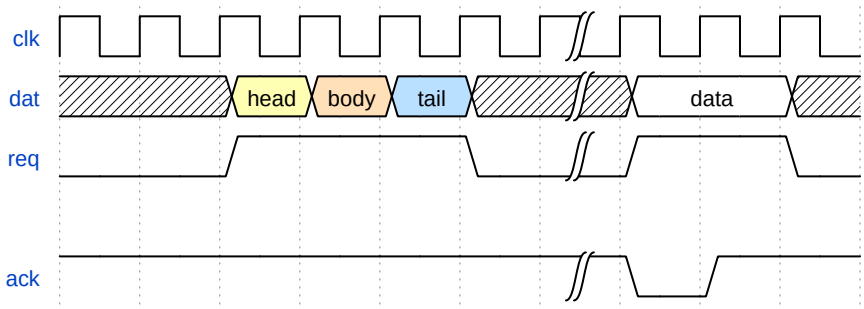


Figure 1. A wavedrom example

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.

A build that incorporates building of Wavedrom diagrams is under development. The build will support making use of the git repo as the single source of truth for a specification.

Until the automation has been developed ans tested, the code for the wavedrom diagrams should be maintained in the repository and the filename for the SVG output should have the same name (with the SVG extension).

2.3.1. Diagram types available

Asciidoctor supports numerous diagram types. Unfortunately while the intention is to support wavedrom diagrams, there are some bugs that prevent implementation of automated generation of them directly from asciidoc files.

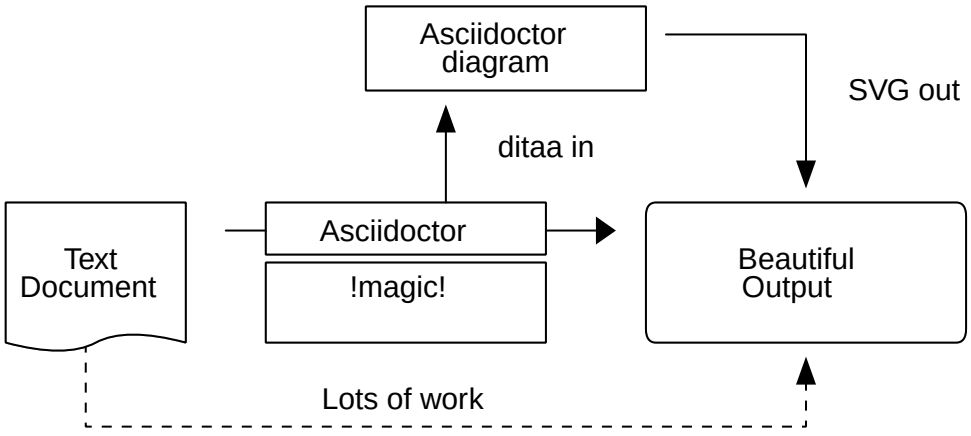
Once the bugs are resolved, wavedrom should work as other diagram types work.

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:

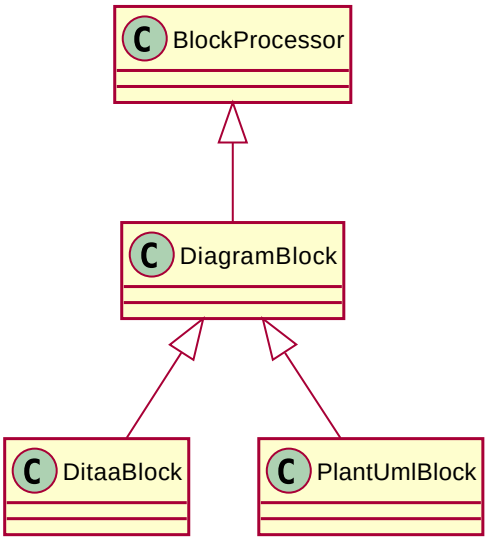


Following is source for a plantuml diagram:

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

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

Which renders to:



2.3.2. Handling Wavedrom diagrams

For the time being and until wavedrom is fully operational as an extension to asciidoctor-diagram, the following workaround is required if you are making use of wavedrom diagrams in your specification:

1. Following guidelines available at wavedrom.com/, create a json-formatted script for your diagram.
2. Using a informative filename, save that script in the `images/wavedrom/diagram_scripts/` relative to your asciidoc source files.
3. Use one of the available wavedrom apps (online, downloadable, or CLI) to generate an `image.svg` file that makes use of the same filename and save it in the `/images` directory.
4. Using the following pattern, reference the `image.svg` file from the asciidoc file:

```
[#wavedrom_example2]
.A second wavedrom example
image::wavedrom/wavedrom-example2.svg[wavedrom_example2]
```

The following json-formatted script is an example that will be used to generate an svg diagram once the automated process is fully implemented.

```
{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'] },
]}
```

Once the diagram file is properly referenced:

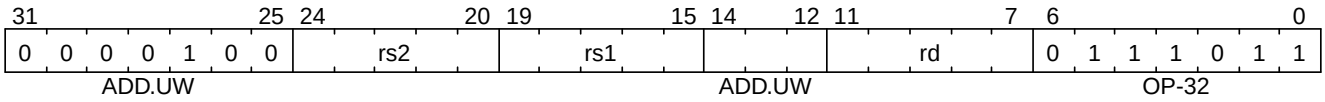


Figure 2. A second wavedrom example

While it appears that there is great interest within the asciidoctor community in enabling Wavedrom diagrams, to date it is not fully integrated with asciidoctor-diagram, which looks for a wavedrom-diagram.app. The downloaded app, however, is intended for use as a manual interface for processing one diagram at a time from individual scripts. For that reason the following output is from an svg that was generated using the Wavedrom CLI.

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/.

Sidebars might be a good solution for some content in the specifications.

```
Text in your document.

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

image::name.png[]

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

Sidebar:

```
.AsciiDoc history
****
AsciiDoc was first released in Nov 2002 by Stuart Rackham.
It was designed from the start to be a shorthand syntax
for producing professional documents like DocBook and LaTeX.
****
```

Rendered:

AsciiDoc history

AsciiDoc was first released in Nov 2002 by Stuart Rackham. It was designed from the start to be a shorthand syntax for producing professional documents like DocBook and LaTeX.

Code blocks

```
x(rd)
```

```

```

3.2. 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	535a63
WARNING	bf6900	9c4d4b
CAUTION	bf3400	c99a2c
IMPORTANT	bf0000	b58f5b

3.3. 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.

```
The hail-and-rainbow protocol can be initiated at 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:

1. double [1]
2. tertiary
3. apocalyptic

A bold statement! [2]

Another outrageous statement.[2]

3.4. 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.

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

Appendix A: Terminology

ABI

Application binary interface. Absctracts and interface for applications to not need to know what lies beneath it in S or M modes.

AEE

Application execution environment—the environment, from bare metal to full operating system, in which an application runs.

AIA

xxx

AIS 31

Information Security service for Europe and the global finance industry (for bank cards), written by BSI.

ALU

Arithmetic Logical Unit

ASIC

Application-Specific Integrated Circuit

AT

xxx

Atomic Layer Deposition

A layer-by-layer process that results in the deposition of thin films one atomic layer at a time in a highly controlled manner.

BF

Brian Float

BFLOAT16

Brain float 16 bit—a vector (V) extension, en.wikipedia.org/wiki/Bfloat16_floating-point_format.

BSI

German Federal Information Security service.

CPU Cache

Many CPUs three kinds of caches to speed up data retrieval: an instruction cache for executable instruction fetch, a data cache for data store and fetch, and a translation lookaside buffer (TLB) for virtual-to-physical address translation for executable instructions and data.

CMOS

Complementary Metal Oxide Semiconductor.

Chemical Vapor Deposition

A chemical deposition process in which the wafer is exposed to one or more volatile precursors, which react and/or decompose on the substrate surface to produce the desired film.

DRAM

Dynamic Random Access Memory

eDRAM

Embedded DRAM

ELEN

Element length

GE

Gate Equivalent

Hart

HARdware Thread—at machine-mode level each hart is a real hardware thread, either one hart per core without hardware multithreading, or multiple harts per core with hardware multithreading, and 'hart' represents the hardware resource. It is possible to emulate harts in software, for example, privileged execution environments can multiplex lesser-privileged harts onto physical hardware using timer interrupts. Note that co-operative multithreading within the same privilege level is not a compliant implementation. Across all implementation choices, we retain the concept of a hart as a resource abstraction representing an independently advancing RISC-V execution context within a RISC-V execution environment.

IC

Integrated Circuit

IRC

The IRC Internet Relay Chat protocol is for use with text based conferencing; the simplest client being any socket program capable of connecting to the server.

IIRC

The International Integrated Reporting Council (IIRC) (previously the International Integrated Reporting Committee). was formed in August 2010 and aims to create a globally accepted framework for a process that results in communications by an organisation about value creation over time.

IMSIC

(Instruction Memory Set ? ?)

ISA

Instruction Set Architecture

MCM

Multi-chip Module

MMU

Memory Management Unit

NAND

Not-and.

NIST

Keeps the standard time for America, defines 1 inch, and also cryptographic standards.

NOR

Logical NOR, known as Pierce's Equivalent, Quine's Dagger, the ampcheck (from the Greek for "cutting both ways"), joint denial, or neither-nor, operates on two logical values, typically from two propositions, that produces a value of true if and only if both operands are false. In other words, it produces a value of false if and only if at least one operand is true.

Photolithography

In microprocessor maufacturing, a process of using light to transfer a geometric pattern from a photomask (also called an optical mask) pattern parts to a photosensitive substrate on a thin film (substrate or wafer). The process can also make use of chemical photoresist on the substrate.

PLIC

Progressive Lossless Image Coding.

PQC

Post-Quantum Cryptography, due to replace RSA and ECC in NIST cryptography [PQC] as well as military [NSA].

PTE

Page Table Entry

PTEP

xxx

PTG.2

A physical random number generator class defined in AIS 31 / CC.

PUD

xxx

QEMU

QEMU (Quick EMUlator) is a free and open-source emulator and virtualizer that can perform hardware virtualization.

RV

Reliability verification is a category of physical verification that helps ensure the robustness of a design by considering the context of schematic and layout information to perform user-definable checks against various electrical and physical design rules that reduce susceptibility to premature or catastrophic electrical failures, usually over time.

Rocket

Parameterized SoC generator written in Chisel, designed to helps tune the design under different performance, power, area constraints, and diverse technology nodes.

SFENCE

Orders processor execution relative to all memory stores prior to the SFENCE instruction. The processor ensures that every store prior to SFENCE is globally visible before any store after SFENCE becomes globally visible. The SFENCE instruction is ordered with respect to memory stores, other SFENCE instructions, MFENCE instructions, and any serializing instructions (such as the CPUID instruction). It is not ordered with respect to memory loads or the LFENCE instruction.

SFENCE.VMA

(instruction wrapper?)

SoC

System on Chip.

SP 800 90B

Used in military & USGOV random security evaluations, written by NIST.

SRAM

Static Random Access Memory

TLB

Translation Lookaside Buffer; enhances speed in retrieving a value by storing a memory address.

VM

Virtual Machine

VMA

(..Virtual Memory Allocation ??)

XLEN

Register width—etymology involves reference to mathematical `X` and abbreviation of the word "length."

ZBT

Zero Bus Turnaround

ZFew

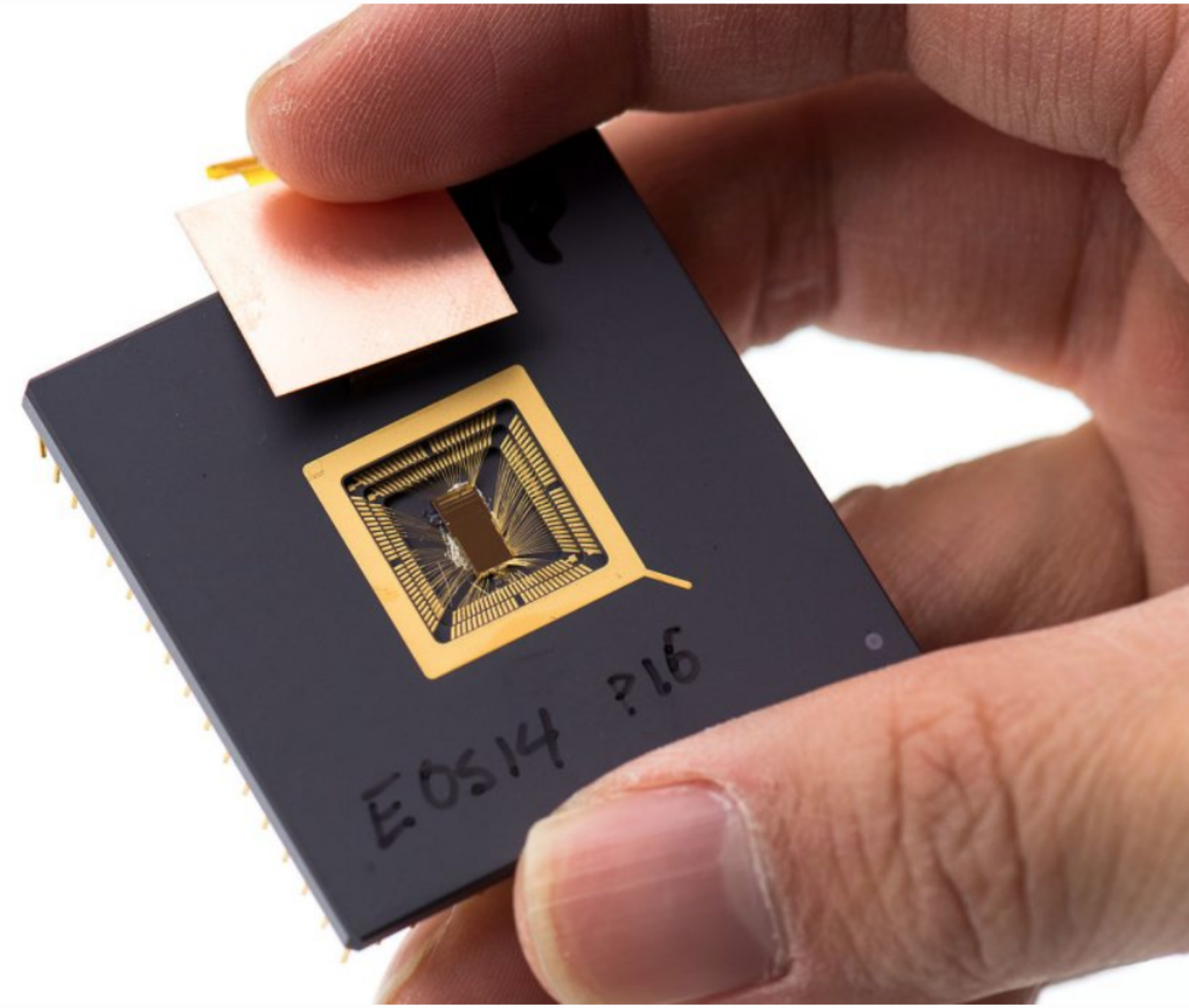
xxxx

Chapter 4. Bibliography

This is a placeholder file while bibliography is being implemented.

Index

K


EOS14 ?16