

# Module Interface Specification for Lattice Boltzmann Solvers

Peter Michalski

November 11, 2019

# 1 Revision History

Date	Version	Notes
Nov. 25, 2019	1.0	Initial Document

## 2 Symbols, Abbreviations and Acronyms

See CA Documentation for Lattice Boltzmann Solvers ([Michalski, a](#)).

# Contents

<b>1</b>	<b>Revision History</b>	<b>i</b>
<b>2</b>	<b>Symbols, Abbreviations and Acronyms</b>	<b>ii</b>
<b>3</b>	<b>Introduction</b>	<b>1</b>
<b>4</b>	<b>Notation</b>	<b>1</b>
<b>5</b>	<b>Module Decomposition</b>	<b>2</b>
<b>6</b>	<b>MIS of M1: Hardware Hiding Module</b>	<b>3</b>
6.1	Module . . . . .	3
6.2	Uses . . . . .	3
6.3	Syntax . . . . .	3
6.3.1	Exported Constants . . . . .	3
6.3.2	Exported Access Programs . . . . .	3
6.4	Semantics . . . . .	3
6.4.1	State Variables . . . . .	3
6.4.2	Environment Variables . . . . .	3
6.4.3	Assumptions . . . . .	3
6.4.4	Access Routine Semantics . . . . .	3
6.4.5	Local Functions . . . . .	4
<b>7</b>	<b>MIS of M2: System Control Module</b>	<b>5</b>
7.1	Module . . . . .	5
7.2	Uses . . . . .	5
7.3	Syntax . . . . .	5
7.3.1	Exported Constants . . . . .	5
7.3.2	Exported Access Programs . . . . .	5
7.4	Semantics . . . . .	5
7.4.1	State Variables . . . . .	5
7.4.2	Environment Variables . . . . .	5
7.4.3	Assumptions . . . . .	5
7.4.4	Access Routine Semantics . . . . .	5
7.4.5	Local Functions . . . . .	6
<b>8</b>	<b>MIS of M3: Input Reading Module</b>	<b>7</b>
8.1	Module . . . . .	7
8.2	Uses . . . . .	7
8.3	Syntax . . . . .	7
8.3.1	Exported Constants . . . . .	7
8.3.2	Exported Access Programs . . . . .	7

8.4	Semantics . . . . .	7
8.4.1	State Variables . . . . .	7
8.4.2	Environment Variables . . . . .	7
8.4.3	Assumptions . . . . .	7
8.4.4	Access Routine Semantics . . . . .	7
8.4.5	Local Functions . . . . .	8
<b>9</b>	<b>MIS of M4: Input Checking Module</b>	<b>9</b>
9.1	Module . . . . .	9
9.2	Uses . . . . .	9
9.3	Syntax . . . . .	9
9.3.1	Exported Constants . . . . .	9
9.3.2	Exported Access Programs . . . . .	9
9.4	Semantics . . . . .	9
9.4.1	State Variables . . . . .	9
9.4.2	Environment Variables . . . . .	9
9.4.3	Assumptions . . . . .	9
9.4.4	Access Routine Semantics . . . . .	9
9.4.5	Local Functions . . . . .	10
<b>10</b>	<b>MIS of M5: LBM Control Module</b>	<b>11</b>
10.1	Module . . . . .	11
10.2	Uses . . . . .	11
10.3	Syntax . . . . .	11
10.3.1	Exported Constants . . . . .	11
10.3.2	Exported Access Programs . . . . .	11
10.4	Semantics . . . . .	11
10.4.1	State Variables . . . . .	11
10.4.2	Environment Variables . . . . .	11
10.4.3	Assumptions . . . . .	11
10.4.4	Access Routine Semantics . . . . .	11
10.4.5	Local Functions . . . . .	12
<b>11</b>	<b>MIS of M6: Streaming Module</b>	<b>13</b>
11.1	Module . . . . .	13
11.2	Uses . . . . .	13
11.3	Syntax . . . . .	13
11.3.1	Exported Constants . . . . .	13
11.3.2	Exported Access Programs . . . . .	13
11.4	Semantics . . . . .	13
11.4.1	State Variables . . . . .	13
11.4.2	Environment Variables . . . . .	13
11.4.3	Assumptions . . . . .	13

11.4.4	Access Routine Semantics . . . . .	13
11.4.5	Local Functions . . . . .	14
<b>12</b>	<b>MIS of M7: Collision Module</b>	<b>15</b>
12.1	Module . . . . .	15
12.2	Uses . . . . .	15
12.3	Syntax . . . . .	15
12.3.1	Exported Constants . . . . .	15
12.3.2	Exported Access Programs . . . . .	15
12.4	Semantics . . . . .	15
12.4.1	State Variables . . . . .	15
12.4.2	Environment Variables . . . . .	15
12.4.3	Assumptions . . . . .	15
12.4.4	Access Routine Semantics . . . . .	15
12.4.5	Local Functions . . . . .	16
<b>13</b>	<b>MIS of M8: Problem Module</b>	<b>17</b>
13.1	Module . . . . .	17
13.2	Uses . . . . .	17
13.3	Syntax . . . . .	17
13.3.1	Exported Constants . . . . .	17
13.3.2	Exported Access Programs . . . . .	17
13.4	Semantics . . . . .	17
13.4.1	State Variables . . . . .	17
13.4.2	Environment Variables . . . . .	17
13.4.3	Assumptions . . . . .	17
13.4.4	Access Routine Semantics . . . . .	17
13.4.5	Local Functions . . . . .	18
<b>14</b>	<b>MIS of M9: Lattice Module</b>	<b>19</b>
14.1	Module . . . . .	19
14.2	Uses . . . . .	19
14.3	Syntax . . . . .	19
14.3.1	Exported Constants . . . . .	19
14.3.2	Exported Access Programs . . . . .	19
14.4	Semantics . . . . .	19
14.4.1	State Variables . . . . .	19
14.4.2	Environment Variables . . . . .	19
14.4.3	Assumptions . . . . .	19
14.4.4	Access Routine Semantics . . . . .	19
14.4.5	Local Functions . . . . .	20

<b>15 MIS of M10: Boundary Module</b>	<b>21</b>
15.1 Module . . . . .	21
15.2 Uses . . . . .	21
15.3 Syntax . . . . .	21
15.3.1 Exported Constants . . . . .	21
15.3.2 Exported Access Programs . . . . .	21
15.4 Semantics . . . . .	21
15.4.1 State Variables . . . . .	21
15.4.2 Environment Variables . . . . .	21
15.4.3 Assumptions . . . . .	21
15.4.4 Access Routine Semantics . . . . .	21
15.4.5 Local Functions . . . . .	22
<b>16 MIS of M11: Output Module</b>	<b>23</b>
16.1 Module . . . . .	23
16.2 Uses . . . . .	23
16.3 Syntax . . . . .	23
16.3.1 Exported Constants . . . . .	23
16.3.2 Exported Access Programs . . . . .	23
16.4 Semantics . . . . .	23
16.4.1 State Variables . . . . .	23
16.4.2 Environment Variables . . . . .	23
16.4.3 Assumptions . . . . .	23
16.4.4 Access Routine Semantics . . . . .	23
16.4.5 Local Functions . . . . .	24
<b>17 Appendix</b>	<b>26</b>

### 3 Introduction

The following document details the Module Interface Specifications for Lattice Boltzmann Solvers, which provides a library of services based on Lattice Boltzmann Methods (LBM). LBM are a family of fluid dynamics algorithms for simulating single-phase and multiphase fluid flows, often incorporating additional physical complexities ([Chen and Doolen, 1998](#)).

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found [here](#) ([Michalski, b](#)).

### 4 Notation

[You should describe your notation. You can use what is below as a starting point. —SS]

The structure of the MIS for modules comes from [Hoffman and Strooper \(1999\)](#), with the addition that template modules have been adapted from [Ghezzi et al. \(2003\)](#). The mathematical notation comes from Chapter 3 of [Hoffman and Strooper \(1999\)](#). For instance, the symbol  $:=$  is used for a multiple assignment statement and conditional rules follow the form  $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | \dots | c_n \Rightarrow r_n)$ .

The following table summarizes the primitive data types used by Lattice Boltzmann Solvers.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in $(-\infty, \infty)$
natural number	$\mathbb{N}$	a number without a fractional component in $[1, \infty)$
real	$\mathbb{R}$	any number in $(-\infty, \infty)$

The specification of Lattice Boltzmann Solvers uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, Lattice Boltzmann Solvers uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.



## 5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	Level 2
Hardware-Hiding Module	M1: Hardware Hiding Module
Behaviour-Hiding Module	M2: System Control Module
	M3: Input Reading Module
	M4: Input Checking Module
	M5: LBM Control Module
	M6: Streaming Module
	M7: Collision Module
	M8: Problem Module
	M9: Lattice Module
Software Decision Module	M10: Boundary Module
	M11: Output Module

Table 1: Module Hierarchy

## 6 MIS of M1: Hardware Hiding Module

[Use labels for cross-referencing —SS]

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L<sup>A</sup>T<sub>E</sub>X for hyperlinks to external documents. —SS]

### 6.1 Module

[Short name for the module —SS]

### 6.2 Uses

### 6.3 Syntax

#### 6.3.1 Exported Constants

#### 6.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

### 6.4 Semantics

#### 6.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

#### 6.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

#### 6.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

#### 6.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]

- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

#### **6.4.5 Local Functions**

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

## 7 MIS of M2: System Control Module

[Use labels for cross-referencing —SS]

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L<sup>A</sup>T<sub>E</sub>X for hyperlinks to external documents. —SS]

### 7.1 Module

[Short name for the module —SS]

### 7.2 Uses

### 7.3 Syntax

#### 7.3.1 Exported Constants

#### 7.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

### 7.4 Semantics

#### 7.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

#### 7.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

#### 7.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

#### 7.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]

- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

#### **7.4.5 Local Functions**

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

## 8 MIS of M3: Input Reading Module

[Use labels for cross-referencing —SS]

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L<sup>A</sup>T<sub>E</sub>X for hyperlinks to external documents. —SS]

### 8.1 Module

[Short name for the module —SS]

### 8.2 Uses

### 8.3 Syntax

#### 8.3.1 Exported Constants

#### 8.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

### 8.4 Semantics

#### 8.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

#### 8.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

#### 8.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

#### 8.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]

- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

#### **8.4.5 Local Functions**

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

## 9 MIS of M4: Input Checking Module

[Use labels for cross-referencing —SS]

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L<sup>A</sup>T<sub>E</sub>X for hyperlinks to external documents. —SS]

### 9.1 Module

[Short name for the module —SS]

### 9.2 Uses

### 9.3 Syntax

#### 9.3.1 Exported Constants

#### 9.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

### 9.4 Semantics

#### 9.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

#### 9.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

#### 9.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

#### 9.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]



- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

#### **9.4.5 Local Functions**

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

## 10 MIS of M5: LBM Control Module

[Use labels for cross-referencing —SS]

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L<sup>A</sup>T<sub>E</sub>X for hypperlinks to external documents. —SS]

### 10.1 Module

[Short name for the module —SS]

### 10.2 Uses

### 10.3 Syntax

#### 10.3.1 Exported Constants

#### 10.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

### 10.4 Semantics

#### 10.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

#### 10.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

#### 10.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

#### 10.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]

- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

#### **10.4.5 Local Functions**

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

## 11 MIS of M6: Streaming Module

[Use labels for cross-referencing —SS]

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L<sup>A</sup>T<sub>E</sub>X for hyperlinks to external documents. —SS]

### 11.1 Module

[Short name for the module —SS]

### 11.2 Uses

### 11.3 Syntax

#### 11.3.1 Exported Constants

#### 11.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

### 11.4 Semantics

#### 11.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

#### 11.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

#### 11.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

#### 11.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]

- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

#### **11.4.5 Local Functions**

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

## 12 MIS of M7: Collision Module

[Use labels for cross-referencing —SS]

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L<sup>A</sup>T<sub>E</sub>X for hyperlinks to external documents. —SS]

### 12.1 Module

[Short name for the module —SS]

### 12.2 Uses

### 12.3 Syntax

#### 12.3.1 Exported Constants

#### 12.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

### 12.4 Semantics

#### 12.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

#### 12.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

#### 12.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

#### 12.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]

- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

#### **12.4.5 Local Functions**

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

## 13 MIS of M8: Problem Module

[Use labels for cross-referencing —SS]

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L<sup>A</sup>T<sub>E</sub>X for hypperlinks to external documents. —SS]

### 13.1 Module

[Short name for the module —SS]

### 13.2 Uses

### 13.3 Syntax

#### 13.3.1 Exported Constants

#### 13.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

### 13.4 Semantics

#### 13.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

#### 13.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

#### 13.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

#### 13.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]



- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

#### **13.4.5 Local Functions**

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

## 14 MIS of M9: Lattice Module

[Use labels for cross-referencing —SS]

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L<sup>A</sup>T<sub>E</sub>X for hypperlinks to external documents. —SS]

### 14.1 Module

[Short name for the module —SS]

### 14.2 Uses

### 14.3 Syntax

#### 14.3.1 Exported Constants

#### 14.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

### 14.4 Semantics

#### 14.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

#### 14.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

#### 14.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

#### 14.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]

- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

#### **14.4.5 Local Functions**

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

## 15 MIS of M10: Boundary Module

[Use labels for cross-referencing —SS]

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L<sup>A</sup>T<sub>E</sub>X for hyperlinks to external documents. —SS]

### 15.1 Module

[Short name for the module —SS]

### 15.2 Uses

### 15.3 Syntax

#### 15.3.1 Exported Constants

#### 15.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

### 15.4 Semantics

#### 15.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

#### 15.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

#### 15.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

#### 15.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]

- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

#### **15.4.5 Local Functions**

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

## 16 MIS of M11: Output Module

[Use labels for cross-referencing —SS]

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L<sup>A</sup>T<sub>E</sub>X for hyperlinks to external documents. —SS]

### 16.1 Module

[Short name for the module —SS]

### 16.2 Uses

### 16.3 Syntax

#### 16.3.1 Exported Constants

#### 16.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

### 16.4 Semantics

#### 16.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

#### 16.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

#### 16.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

#### 16.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]

- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

#### **16.4.5 Local Functions**

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

## References

- Shiyi Chen and Gary D Doolen. Lattice Boltzmann method for fluid flows. *Annual review of fluid mechanics*, 30(1):329–364, 1998.
- Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.
- Daniel Hoffman and Paul Strooper. Software Design, Automated Testing, and Maintenance—A Practical Approach. 1999.
- Peter Michalski. Lattice Boltzmann Solvers - CA, a. URL <https://github.com/peter-michalski/LatticeBoltzmannSolvers/blob/master/docs/SRS/CA.pdf>.
- Peter Michalski. Lattice Boltzmann Solvers, b. URL <https://github.com/peter-michalski/LatticeBoltzmannSolvers>.



## 17 Appendix

[Extra information if required —SS]