Module Interface Specification for FFT Library

Yuzhi Zhao

November 26, 2017

1 Revision History

Date	**Version**	**Notes**
Date 1	1.0	Notes
Date 2	1.1	Notes

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at https://github.com/741ProjectFFT/FFT/tree/master/Doc/SRS

[Also add any additional symbols, abbreviations or acronyms —SS]

Contents

1 Revision History i

2 Symbols, Abbreviations and Acronyms ii

3 Introduction 1

4 Notation 1

5 Module Decomposition 1

6 MIS of Input Computing Module 3
- 6.1 Module . . . 3
- 6.2 Uses . . . 3
- 6.3 Syntax . . . 3
  - 6.3.1 Exported Access Programs . . . 3
- 6.4 Semantics . . . 3
  - 6.4.1 State Variables . . . 3
  - 6.4.2 Access Routine Semantics . . . 3

7 MIS of FFT Calculation Module 4
- 7.1 Module . . . 4
- 7.2 Uses . . . 4
- 7.3 Syntax . . . 4
  - 7.3.1 Exported Access Programs . . . 4
- 7.4 Semantics . . . 5
  - 7.4.1 State Variables . . . 5
  - 7.4.2 Access Routine Semantics . . . 5

8 MIS of Output Computing Module 6
- 8.1 Module . . . 6
- 8.2 Uses . . . 6
- 8.3 Syntax . . . 6
  - 8.3.1 Exported Access Programs . . . 6
- 8.4 Semantics . . . 6
  - 8.4.1 State Variables . . . 6
  - 8.4.2 Access Routine Semantics . . . 6

9 Appendix 7

3 Introduction

The following document details the Module Interface Specifications for [Fill in your project name and description —SS]

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at .... [provide the url for your repo —SS]

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 ?, with the addition that template modules have been adapted from ?. The mathematical notation comes from Chapter 3 of ?. For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | ... | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by Program Name.

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

The specification of Program Name 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, Program Name 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	
Behaviour-Hiding	Input Computing Module FFT Calculation Module Output Computing Module
Software Decision	Array Data Structure Module

Table 1: Module Hierarchy

6 MIS of Input Computing Module

[Use labels for cross-referencing —SS]

6.1 Module

input_compute

6.2 Uses

None

6.3 Syntax

6.3.1 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
inputCmpt	string, $\mathbb{Z} \in \{2,3\}$, string	-	
filename	string	-	-
radix	$\mathbb{Z} \in \{2,3\}$	-	-
data_type	string	-	-

6.4 Semantics

6.4.1 State Variables

None

6.4.2 Access Routine Semantics

load_data():

- transition: None
- output: None
- exception: exc := (FilenameCannotFound, FileReadFail)

verify_para():

- transition: None
- output: None

- exception: exc := (
  radix $\notin \{2, 3\} \Rightarrow$ RadixIsNotValid,
  data_type $\neq$ Para_data_type $\Rightarrow$ ParaNotMatchRealDataType)

verify_data():

- transition: None
- output: None
- exception: exc := (
  (radix $= 2$, numberOfData $\neq 2^n, n \in \mathbb{N} \Rightarrow$ NumberOfDataNotMatchRadixConstraint)
  (radix $= 3$, numberOfData $\neq 3^n, n \in \mathbb{N} \Rightarrow$ NumberOfDataNotMatchRadixConstraint))

fill_zero():

- transition: None
- output: None
- exception: exc := None

7 MIS of FFT Calculation Module

[Use labels for cross-referencing —SS]

7.1 Module

FFT_calculate

7.2 Uses

None

7.3 Syntax

7.3.1 Exported Access Programs

Name	In	Out	Exceptions
r2com()	$\mathbb{C}^n$	-	
r2real()	$\mathbb{R}^n$	-	
r3com()	$\mathbb{C}^n$	-	
r3real()	$\mathbb{R}^n$	-	

7.4 Semantics

7.4.1 State Variables

None

7.4.2 Access Routine Semantics

rearrange_data_sequence():

- transition: None
- output: None
- exception: exc := None

calculate_ω_N_set():

- transition: None
- output: None
- exception: exc := None

radix2_multiply_calculation():

- transition: None
- output: None
- exception: exc := None

radix2_plus_calculation():

- transition: None
- output: None
- exception: exc := None

radix3_multiply_calculation():

- transition: None
- output: None
- exception: exc := None

radix3_plus_calculation():

- transition: None
- output: None
- exception: exc := None

8 MIS of Output Computing Module

[Use labels for cross-referencing —SS]

8.1 Module

output_compute

8.2 Uses

None

8.3 Syntax

8.3.1 Exported Access Programs

Name	In	Out	Exceptions
outputCmpt	$\mathbb{C}^n$ or $\mathbb{R}^n$	-	

8.4 Semantics

8.4.1 State Variables

None

8.4.2 Access Routine Semantics

generate_file():

- transition: None
- output: None
- exception: exc := (
  GenerateFileFail)

verify_output_data():

- transition: None
- output: None
- exception: exc := (
  outputDataNumber $\neq$ inputDataNumberAfterFill0 $\Rightarrow$ DataNumberError)

9 Appendix

[Extra information if required —SS]