Module Interface Specification for FFT Library

Yuzhi Zhao

December 19, 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

[You don't actually seem to have used any symbols from your SRS. Also, the symbol list in your SRS does not seem to have been updated. —SS]

[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 **2**

6 MIS of Input 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 . . . 4

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

8 MIS of Output Module **7**
- 8.1 Module . . . 7
- 8.2 Uses . . . 7
- 8.3 Syntax . . . 7
  - 8.3.1 Exported Access Programs . . . 7
- 8.4 Semantics . . . 7
  - 8.4.1 State Variables . . . 7
  - 8.4.2 Access Routine Semantics . . . 7

9 Appendix **9**

3 Introduction

The following document details the Module Interface Specifications for FFT (Fourier Transform Library). This library provides radix 2 and radix 3 FFT and IFFT functions. The input values will be read from files and the output will be written into a new text file. Fast Fourier transforms are widely used for many applications in engineering, science, and mathematics. FFT's importance derives from the fact that in signal processing and image processing it has made working in frequency domain equally computationally feasible as working in temporal or spatial domain. So this library will provide good service to other programs.

[You really should figure out how to do 80 column wide text. This will become important for the code and documentation that you add to Drasil. —SS]

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/741ProjectFFT.

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 (-∞, ∞)
complex	$\mathbb{C}$	any number can be expressed in the form a + bi, where a and b are real numbers
pointer	*	a programming language object whose value refers to another value stored elsewhere in the computer memory using its memory address.
boolean	$\mathbb{B}$	a programming language object whose value refers to another value stored elsewhere in the computer memory using its memory address.

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 Module FFT Calculation Module Output Module
Software Decision	Array Data Structure Module

Table 1: Module Hierarchy

6 MIS of Input Module

[Use labels for cross-referencing —SS]

6.1 Module

Load

6.2 Uses

None

6.3 Syntax

6.3.1 Exported Access Programs

Name	In	Out	Exceptions
load_data_complex	$char^N(N \in \mathbb{N})$, $\mathbb{R} * *$, $\mathbb{R} * *$, $\mathbb{N}*$	-	
load_data_real	$char^N(N \in \mathbb{N})$, $\mathbb{R} * *$, $\mathbb{N}*$	-	
makeComplexArray	$\mathbb{R}*$, $\mathbb{R}*$, $\mathbb{N}$	$\mathbb{C}*$	
filename	$char^N(N \in \mathbb{N})$	-	-
ppRealNumber	$\mathbb{N} * *$	$\mathbb{N} * *$	-
ppImageNumber	$\mathbb{N} * *$	$\mathbb{N} * *$	-
pInputSize	$\mathbb{Z}*$	$\mathbb{Z}*$	-
size	$\mathbb{N}$	-	-
realArray	$\mathbb{R}*$	-	-
imgArray	$\mathbb{R}*$	-	-

6.4 Semantics

6.4.1 State Variables

ppRealNumber: $\mathbb{N} * *$
ppImageNumber: $\mathbb{N} * *$
pInputSize: $\mathbb{Z}*$
realArray: $\mathbb{R}*$

imgArray: $\mathbb{R}*$

6.4.2 Access Routine Semantics

load_data_complex :

- transition: ppRealNumber, ppImageNumber, pInputSize
- output: *out* := self
- exception: exc := (FileReadFail)

load_data_real :

- transition: ppRealNumber, pInputSize
- output: *out* := self
- exception: exc := (FileReadFail)

verify_number_input():

- transition: Check whether pInputSize = $2^{\mathbb{N}}$ or = $3^{\mathbb{N}}$
- output: warning message
- 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))

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
radix2FFTCalc	$\mathbb{C}^n$,		
	$\mathbb{N}$		
	$\mathbb{N}$	$\mathbb{C}^n$,	
radix3FFTCalc	$\mathbb{C}^n$,		
	$\mathbb{N}$		
	$\mathbb{N}$	$\mathbb{C}^n$,	
fakeImageTerm	$\mathbb{N}$	$\mathbb{R}$	
arrComN	$\mathbb{C}^n$,	$\mathbb{C}^n$,	
size	$\mathbb{N}$	-	
mark	$\mathbb{N}$	-	

7.4 Semantics

[Your syntax and semantics sections do not match. —SS]

7.4.1 State Variables

arrComN: $\mathbb{C}^n$,

7.4.2 Access Routine Semantics

getEvenOddArray(array, size):

- transition: Splite an array to a structure called EVEN_ODD_ARRAY which basically splite an array to new arrays by even and odd order and store them.
- output: a structure with sorted number
- exception: exc := None

getThreeArray(array, size):

- transition: Splite an array to a structure called THREE_ARRAY which basically splite an array to new arrays by three order and store them.
- output: a structure with sorted number
- exception: exc := None

unitBF():

- transition: A radix 2 unit butterfly calculation

- output: A BTRValue structure with two values in it
- exception: exc := None

unitBF2():

- transition: A radix 3 unit butterfly calculation
- output: A BTRValue2 structure with three values in it
- exception: exc := None

ReOrder():

- transition: Decompose the original input sequence to two sequences which were taken from all the odd orders and even orders. And then redecompose each of those two new sequences into another two sequences by picking out the odd terms and the even terms. This function cannot complete the rearranging work untile each subsequence only contains two terms.

- output: new array in new order
- exception: exc := None

ReOrder2():

- transition: Decompose the original input sequence to three sequences which were taken from every three tuples sorted by first, second and third order. And then redecompose each of those three new sequences into another three sequences by picking out the different orded terms. This function cannot complete the rearranging work untile each subsequence only contains three terms.

- output: new array in new order
- exception: exc := None

[The calculate module should show the relevant equations from the SRS, along with your pseudo code description of the behaviour. —SS]

8 MIS of Output 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

[If there are files, then you should have an environment variable. —SS]

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)

[It looks like you ran out of time for finishing this MIS. If you have any questions, please let me know. —SS]

[Two references are missing from your bib file. —SS]

9 Appendix

[Extra information if required —SS]

[If you don't have an appendix, then you don't need this section. —SS]