Module Interface Specification for FFT Library

Yuzhi Zhao

November 29, 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

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.

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 (-∞, ∞)

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))

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_Radix2_data_sequence():

- transition: None
- output: None
- exception: exc := None
- description: 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 anthor 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.
- example: input sequence: 0 1 2 3 4 5 6 7 $\rightarrow$ 0 2 4 6 1 3 5 7 $\rightarrow$ 0 4 2 6 1 5 3 7

rearrange_Radix2_data_sequence():

- transition: None
- output: None
- exception: exc := None
- description: Decompose the original input sequence to three sequences which were taken from all the 1n orders, 2n and 3n orders. And then redecompose each of those three new sequences into anthor three sequences by picking out the 1n terms 2n term and the 3n terms. This function cannot complete the rearranging work untile each subsequence only contains three terms.
- example: input sequence: 0 1 2 3 4 5 6 7 8 $\rightarrow$ 0 3 6 1 4 7

calculate_ω_N^k_set():

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

unit_butterfly_radix2_calculation():

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

- description: This function basically takes two input values and then gives two output values with a butterfly calculation. The rule of butterfly calculation is:
  output_value_1 = input_value_1 + ω_N^k * input_value_2;
  output_value_2 = ω_N^k * input_value_2 - input_value_1

unit_butterfly_radix3_calculation():

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

complex_number_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]