Journal of KOSSE. (2021. 12) Vol. 17, No. 2 pp. 37-48
DOI: https://doi.org/10.14248/JKOSSE.2021.17.2.037
www.kosse.or.kr

ISSN (print) : 1738-480X
ISSN (online) : 2288-3592

Development of an Intellectual Property Core for Floating Point Calculation for Safety Critical MMIS

Nelson Josephat Mwilongo[1)], Jae Cheon Jung[1)]

1) Department of NPP Engineering, KEPCO International Nuclear Graduate School (KINGS)

Abstract: Improving the plant protection system against unforeseen changes/transients during operation is essential to maintain plant safety. Under this condition, it requires rapid and accurate signal processing. The use of an Intellectual Property (IP) core for floating point calculations for Safety Critical MMIS can make numerical computations easier and more precise, improving system accuracy. It can represent and manipulate rational numbers as well as a much broader range of values with dynamic range in nuclear power plant. Systems engineering approach (SE) is used through the development process, it helps to reduce complexity and avoid omissions and invalid assumptions as delivers a better understanding of the stakeholders needs. For the implementation on the FPGA target board, the 32−bit floating−point arithmetic with IEEE−754 standards has designed using Simulink model in Matlab for all operations of addition, subtraction, multiplication and division and VHDL code generated.

Key Words : Safety Critical MMIS, Floating Point Calculation, Intellectual Property Core, Systems Engineering, VHDL code.

Received: October 15, 2021 / **Revised**: December 9, 2021 / **Accepted**: December 13, 2021

* 교신저자 : Jae Cheon Jung/KEPCO International Nuclear Graduate School/jcjung@kings.ac.kr

1. Introduction

Safety Critical Man Machine Interface Systems (MMIS) in Nuclear Power Plant failure or malfunction may severely harm people lives, environment or equipment.[1] Field programmable gate arrays (FPGA) provide solution hardware[2] for application specific computation design for Safety Critical MMIS. This led much study in recent years to focus on floating point implementation on FPGA based systems. It is a challenging problem because floating point numbers require more field than fixed point numbers and availability of physical resources on FPGA (memory, gate) is limited.[3]

The previous approach has been usage of fixed-point arithmetic, which has limitations since it is not always possible to have a bounded enough dynamic range analysis and accuracy verification. It is not possible to obtain a fixed-point solution in those cases where the dynamic range easily adjusted by means of exponent. In nuclear power plant where there are various transients during operation as dynamic range[1] thus floating-point calculation are suitable.

There are several ways of presenting floating point such as Microsoft Binary Format (MBF), Hexadecimal Floating Point (HFP), Min flow (Bfloat16 format) and IEEE-754 floating. However, each has advantages and drawbacks.

Microsoft Binary Format (MBF) is format for floating-point numbers, which used in Microsoft's BASIC language products, including MBASIC, GW-BASIC. It is limited only for Microsoft Basic language products.

Hexadecimal Floating Point (HFP is a format for encoding floating-point numbers on the IBM System/360 computers and supported on machines based on that architecture as well as machines which were intended to be application-compatible with System/360.[3] This format does not define exception conditions such as overflow, underflow, invalid operation, division by zero and inexact result condition.

Min Flow also known as Brain Floating Point 16 bits is a representation of floating point numbers with use in accelerating Machine Learning Inference performance since has wide range presenting with less accuracy.[3]

The IEEE Standard for Floating-Point Arithmetic (IEEE 754)[4] addressed a number of issues that made it difficult to use floating-point implementations reliably and portably. The standard defines arithmetic formats, Interchange formats, rounding rules, operations and exceptional handling such as overflow, underflow, invalid operation, division by zero and inexact result condition. This format has higher precision for floating point calculation than others.

Apart from the existing methods, there is still a problem of complexity during development of Safety Critical MMIS. This leads to difficulties in meeting the compliance with customer needs, system operational environments and other interfacing systems.

In comparison to the existing method of fixed-point calculation, this paper focus on development of Intellectual Property (IP) core for floating point calculation for MMIS based on the IEEE 754 binary standard using Systems Engineering approach. This helps to reduce complexity, avoid omissions and invalid

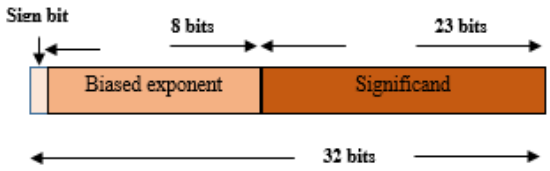
assumptions during development[17] to meet those required compliance.

In nuclear power plant where safety is critical and there are unforeseen changes during operation; this paper helps to improve the plant protection system against those transients using an IP core for floating point calculation where the dynamic range are easily adjusted by means of exponent thus improve the system accuracy as well as plant protection system. Figure 2 shows the IEEE 754 single precision binary format representation[4] with one-bit sign (S), an eight-bit exponent (E) and twenty-three bit fraction (M) or Mantissa.

S: Sign - 1 bit with 1 or 0 for negative or positive number respectively.

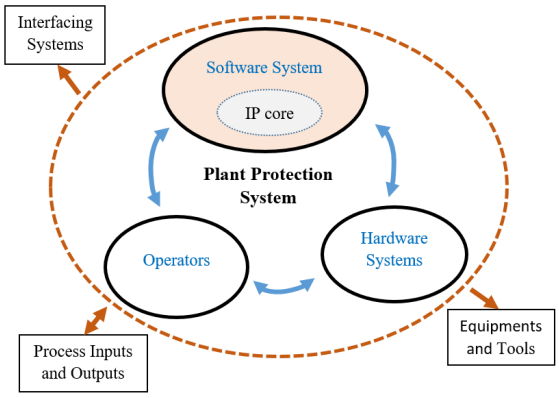
E: Exponent - 8 Bit

M: Mantissa - 23 bits Fraction



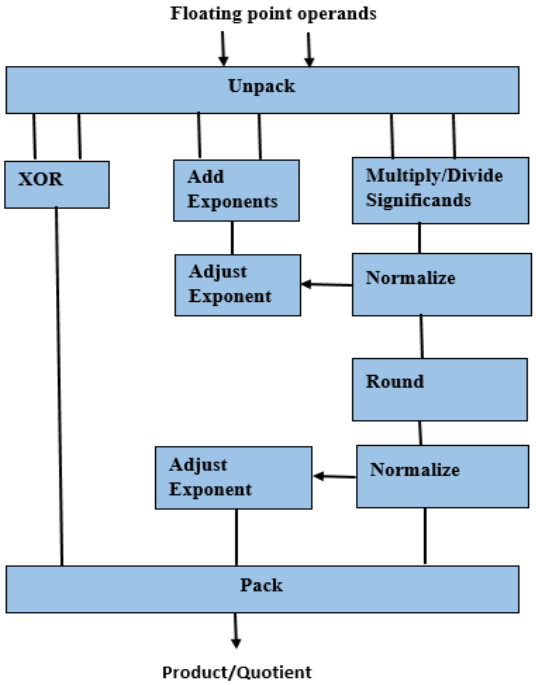
[Figure 1] IEEE 754 Single precision binary format representation

The boundary of this work[5] is stablished inside the plant protection system as showed in Figure 3. Software Systems, Hardware Systems and Operators combine the plant protection system. This work is focusing on the development an IP core for floating point calculation, which is part of Software Systems as shown in Figure 3.



[Figure 2] System Boundary

The operation concept of the system[5] includes unpacking the 32 binary bits to 1-bit for sign, 8-bits for exponent and 23-bits for mantissa or fraction bits[4] and perform arithmetic operation as shown in Figure 4. Below for multiplication or division.



[Figure 3] Operation concept

Main goals pointed out in this work include To develop an Intellectual Property core for

floating point calculation for Safety Critical MMIS.

To improve system accuracy and safety in NPP for unpredictable changing conditions.

Systems engineering approach (SE) is applied here to guide the engineering work of such complex system and enable realization of successful system[5] to implement floating point calculation on FPGA based systems.

2. Systems Engineering Approach

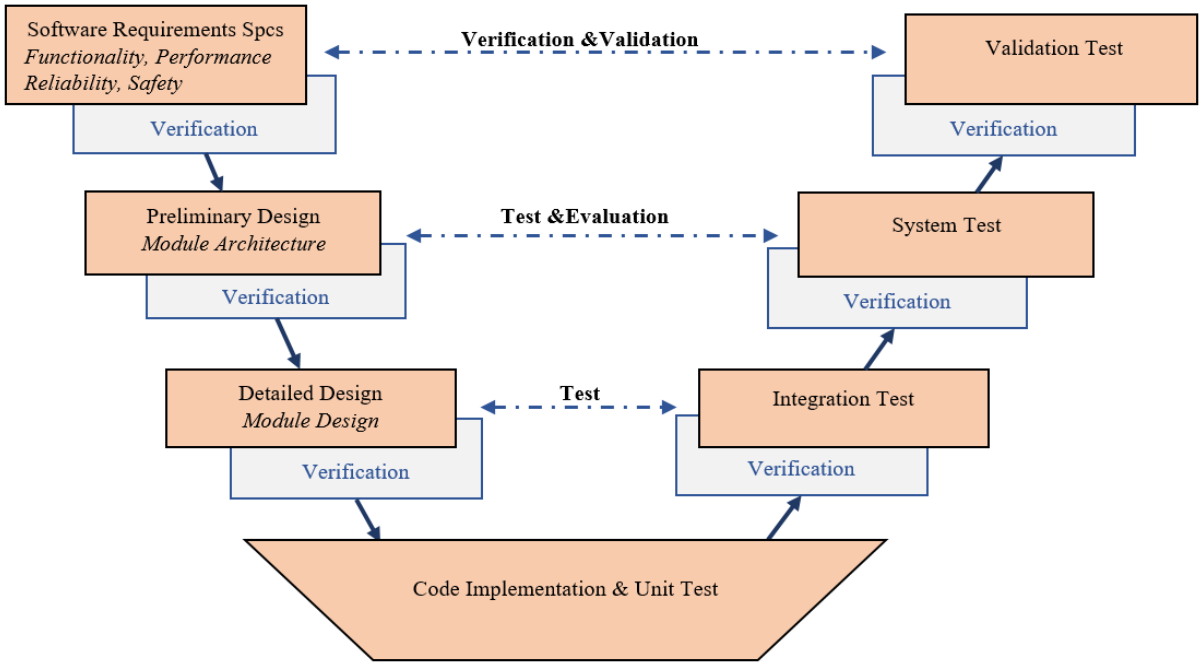
The SE approach points that for a complex system succeed is necessary to start with a problem definition, then stablishing stakeholder's requirements[5] and concept of operation with system boundaries as met in introduction part and finally perform the system verification and validation.[5] The systems engineering V-model[6] used for stepwise development of the proposed floating-point calculation system as shown in Figure 3 below.

2.1 Software Requirements Specifications.

Software Requirements Specifications define those requirements for the software developed from stakeholders' needs. It should be verifiable[7] with person or machine to check whether the software meets the requirements or not.[8] The software requirements also involves analyzing and documenting the requirements for the software.

2.1.1 External interfaces requirements

This describes all inputs into and outputs from the floating-point calculation software system. It includes source of input or destination of output, valid range and accuracy [8], unit of measure, timing and relationships to other inputs/outputs.



[Figure 4] Systems Engineering V- Module

2.1.2 Function requirements.

This requires the system to perform the fundamental action[8] of calculating floating point in accepting and processing the inputs and in processing and generating outputs with high level of accuracy in particular response time[8], different modes and the corresponding conditions of transition including power on and initialization. It interfaces and interacts with the environment[9] including roles, data formats, ranges and constraints of inputs and outputs.

2.1.3 Performance requirements

The system should calculate floating point with high level of accuracy, efficient[9] and high speed.

2.1.4 Design constraints

Design constraints for floating-point calculation software system includes I/O devices, user interface equipment, external interface of the system, performance criteria [9] and programing language used.

2.1.5 Reliability requirements

The system should have high probability[9] to calculate floating point correctly during a specific time duration of operation without maintenance. It should also not affected with any failure of other systems.

2.1.6 Safety requirements

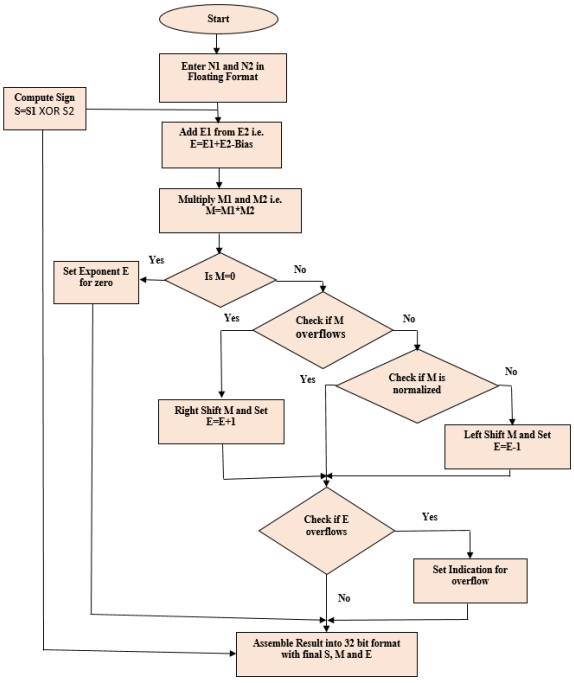
System should be optimized[8] to safety within the constraints of floating point calculation effectiveness, time and cost.

2.1.7 Requirement analysis

Requirement analysis of the software requirements is performed to evaluate each safety critical software requirement with respect to a set of requirement qualities.[9] The requirement qualities defined in this case are completeness, correctness, consistency and testability.

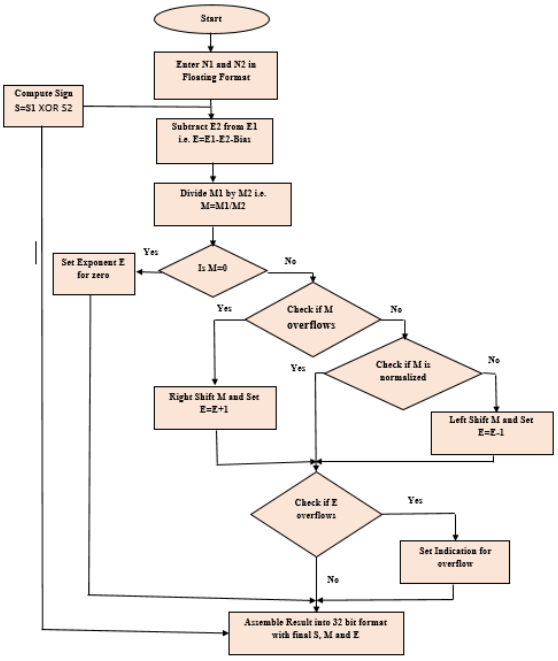
2.2 Preliminary Design

The main purpose of preliminary design is to determine the structure[10] of the software module architecture for floating point calculation. The software module architecture represents the actual software modules, which developed in Matlab Simulink, and then HDL codes generated. Software requirements are allocated to the software module and each requirement should be met by at least one



[Figure 5] Floating point Multiplication flow chart

software module[11]. The software module architecture also contain the interfaces[8] between modules. The software architecture module shown in figure 5 and 6 below for multiplication and division respectively. Let N1 and N2 be normalized operands represented by S1, M1, E1 and S2, M2, E2 as sign bit, mantissa and exponent respectively.



[Figure 6] Floating point Division flow chart

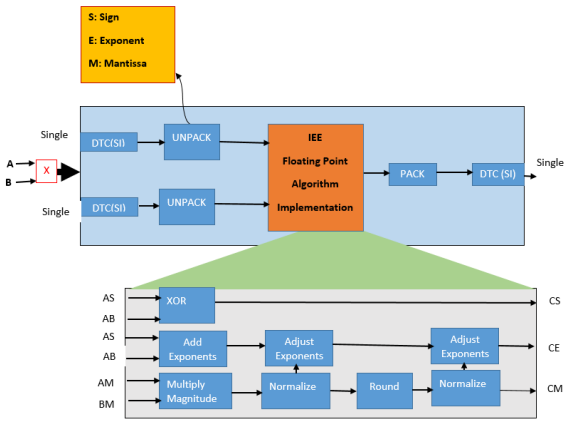
2.3 Detailed design

Detailed design provides sufficient information[12] for the software modules of floating point calculation to ensure ease development and interface implementation. The Matlab Simulink blocks used to design the internal floating-point calculation data structure and generate HDL as language statements. The formats and protocols which interact with external systems for communication[13] are determined and the final design look and feel for human interface displays is developed. The unit test plans for each floating-point calculation module[10] and module interface are developed.

<Table 1> Detailed Design Products

Product	Activity
Floating point calculation Module Design	Detailed Design Analysis
Floating point calculation Module Interfaces	Preliminary Design Analysis
External Interfaces	Detailed Design Analysis
Requirement Traceability	Requirement Management
Unit Test Plan	Test Plan Development
Display Interfaces	Display Development

Floating-point calculation designed to perform the required operation[3] as shown in Figure 7 below for multiplication. It involves unpacking bits to sign, exponent and mantissa [4] as input to the IEEE Floating Point algorithm in Matlab Simulink.



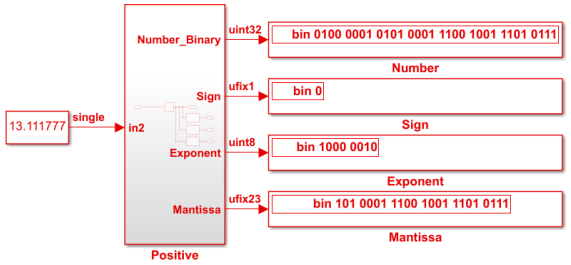
[Figure 7] Floating Point Multiplication

2.4 Code Implementation

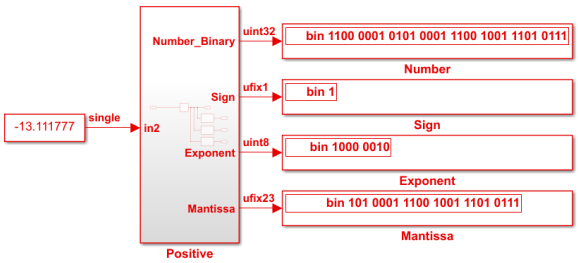
The implementation include development of Matlab Simulink for floating-point calculation and HDL codes generated. As product of

detailed design[10], each floating point calculation module[13] has its design for external interfaces, internal floating point data structure in Matlab Simulink block code. The developed Matlab Simulink reviewed to check the compliance with the floating-point calculation[3] constraints and inspected for logic and control structure.

When the Simulink Block corrected for any defects and compiles with no errors, the HDL code generated. The design of number in IEEE binary format for positive and negative number [4] are show in Figure 8 and 9 below respectively.

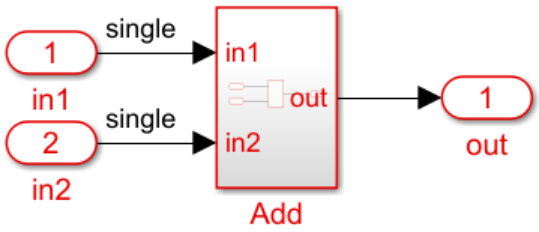


[Figure 8] Positive number in IEEE-754 floating standard



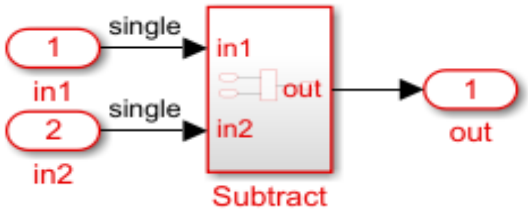
[Figure 9] Negative number in IEEE-754 floating standard

Floating point calculation for addition, subtraction, multiplication and division developed and its HDL code generated as shown in Figures 10, 11, 12 and 13 below respectively



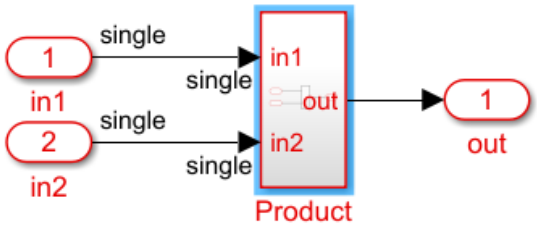
[Figure 10] Addition

```
ENTITY nfp_add_single IS
    PORT(clk            : IN std_logic;
         reset          : IN std_logic;
         enb            : IN std_logic;
         nfp_in1        : IN
std_logic_vector(31 DOWNTO 0);
         nfp_in2        : IN
std_logic_vector(31 DOWNTO 0);
         nfp_out        : OUT
std_logic_vector(31 DOWNTO 0)
 );
    END nfp_add_single;
```



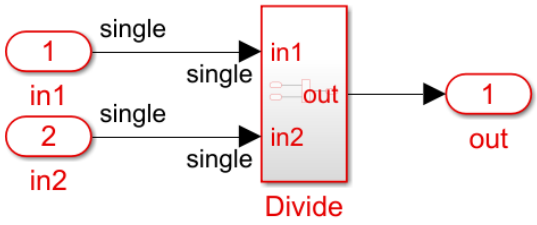
[Figure 11] Subtraction

```
ENTITY nfp_sub_single IS
    PORT(clk            : IN std_logic;
         reset          : IN std_logic;
         enb            : IN std_logic;
         nfp_in1        : IN
std_logic_vector(31 DOWNTO 0);
         nfp_in2        : IN
std_logic_vector(31 DOWNTO 0);
         nfp_out        : OUT
std_logic_vector(31 DOWNTO 0)
 );
    END nfp_sub_single;
```



[Figure 12] Multiplication

```
ENTITY nfp_mul_single IS
     PORT(clk              : IN std_logic;
          reset            : IN std_logic;
          enb              : IN std_logic;
          nfp_in1          : IN
std_logic_vector(31 DOWNTO 0);
          nfp_in2          : IN
std_logic_vector(31 DOWNTO 0);
          nfp_out          : OUT
std_logic_vector(31 DOWNTO 0)
 );
     END nfp_mul_single;
```

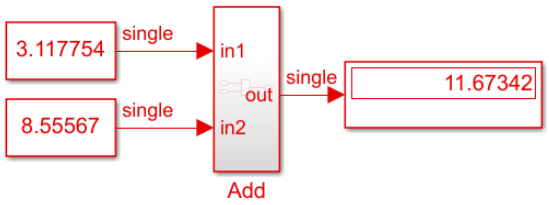


[Figure 13] Division

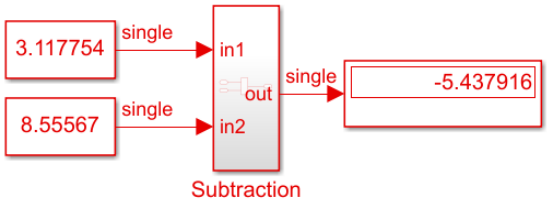
```
ENTITY nfp_div_single IS
     PORT(clk              : IN std_logic;
          reset            : IN std_logic;
          enb              : IN std_logic;
          nfp_in1          : IN
std_logic_vector(31 DOWNTO 0);
          nfp_in2          : IN
std_logic_vector(31 DOWNTO 0);
          nfp_out          : OUT
std_logic_vector(31 DOWNTO 0)
 );
     END nfp_div_single;
```

2.5 Unit Test

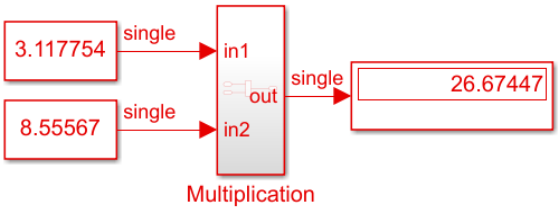
Unit test involve testing independently of individual units implemented to check and identify internal errors[7] such as logic path or floating point numerical algorithms and correct functioning[6] with external interfaces. The unit test done for each floating-point operation as shown in Figures below.



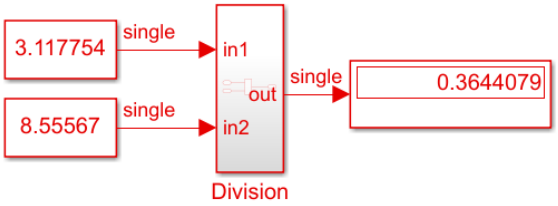
[Figure 14] Floating-point addition



[Figure 15] Floating-point subtraction



[Figure 16] Floating-point multiplication

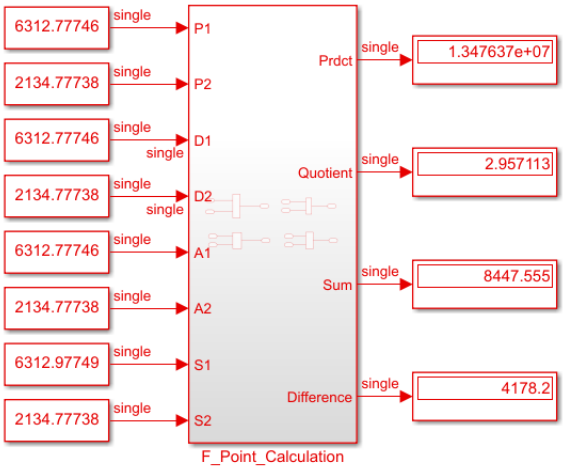


[Figure 17] Floating-point division

2.6 Integration Test

Integration test intends to form increasingly larger sets of integrated[7] floating point

calculation units and test the system to ensure that it is functionally cohesive and operationally correct as collective sets of modules. Whenever necessary external interfaces developed to support he integration.[8]

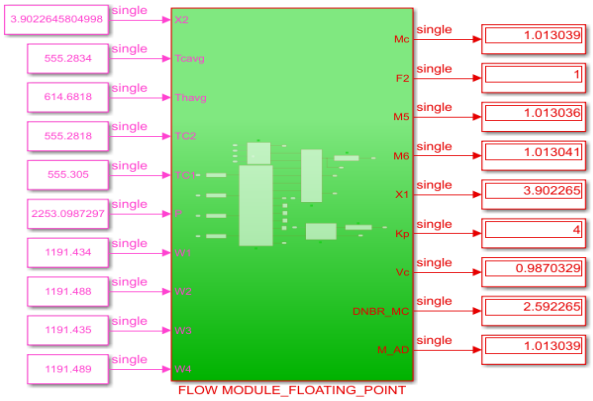


[Figure 18] Floating Point Calculation

2.7 System Test

System testing performed with floating point calculation software modules integrated into a collective and cohesive set.[15] The system were done for floating point calculation for the CPCS[14] flow module. The CPCS flow module[16] simulated to run continuously with input data from KINGS APR1400 Simulator. Flow module deals with computation[18] of

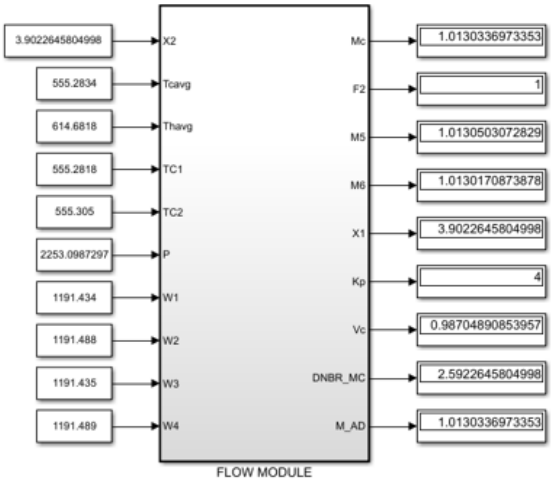
- Pump speed input treatment and pump running detection.
- Specific volumes computation.
- Core flow computation.
- DNBR pump dependent calculation and DNBR margin as shown in Figure 19 below



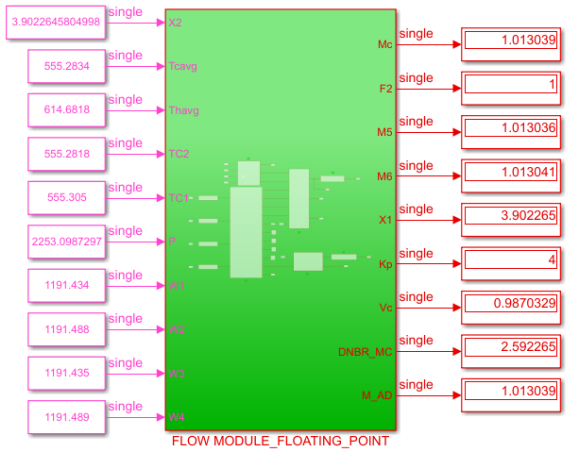
[Figure 19] Flow Module Floating Point Calculation at 20s

2.8 Validation

Validation process[7] performed by CPCS flow module. The flow module developed in Matlab Simulink for normal block with fixed-point calculation and Simulink block with IP core for floating point calculation. The same inputs applied for both blocks and outputs recorded in Table 2.



[Figure 20] CPCS Flow Module



[Figure 21] CPCS Flow Module with IP core for Floating point Calculation.

<Table 2> CPCS Flow module outputs

Parameter	Floating Point Flow Module Output	Normal Flow Module Output	Difference
Mc	1.013039	1.0130336973353	5.30266470000207E-06
F2	1	1	0
M5	1.013036	1.0130503072829	-1.430728289997 24E-05
M6	1.013041	1.0130170873878	2.391261220013 87E-05
X1	3.902265	3.9022645804998	4.19500199910772E-07
Kp	4	4	0
Vc	0.9870329	0.98704890853957	-1.60085395699561E-05
DNBR_MC	2.592265	2.5922645804998	4.19500199910772E-07
M_AD	1.013039	1.0130336973353	5.30266470000207E-06

<Table 3> Output deviation between Flow Module with Floating Point Calculation and Reference data from KEPCO E&C

Input			Output		Reference KEPCO E&C	Deviation
1			Mc	0.787795	0.787794	0.00%
2			F2	13	13	0.00%
3	Thavg	615	M5	1.02146	1.02146	0.00%
4	Vc1	0.980038	M6	0.554125	0.554125	0.00%
5	X2	11.32	X1	11.32	11.32	0.00%
6			Kp	3	3	0.00%
7	P	2250	Vc	0.986709	986709	0.00%
8	F1	1	DNBR_MC	10.01	10.01	0.00%

The table 2 shows the output deviation between the two CPCS flow module[20] developed. However, comparing the result of each Simulink Flow Module with reference data from KEPCO E&C.[14],[19] The Flow module with IP core for floating point calculation gives accurate output with deviation 0.0% as shown in Table 3 above because it involves calculation numbers between integers (fractional numbers) hence improve the system accuracy for dynamic range as well as plant protection system against unforeseen/ unpredictable changes during operation.[6]

3. Conclusion

This paper introduced a systems engineering (SE) approach to develop an Intellectual Property (IP) core for Floating Point Calculation for Safety Critical MMIS. Due to its complexity, SE approach looks the whole system not only engineering design of system but also external factors which significantly constrain the design. Moreover it avoid omissions and invalid assumptions during development by defining clear the system boundary of safe state and system requirements to ensure compliance with customer needs, system operational environment and interfacing systems. The systems engineering V-model used for step wise development of the proposed floating point calculation system by addressing the system requirement specifications and balances both development and testing activities.

The concept of floating point calculation for Safety Critical MMIS in nuclear power plant general introduced to improve the plant protection against unforeseen/unpredictable changes during plant operation due to precise calculation of Floating point compared with

fixed points. The floating-point calculation involves numbers between integers hence improve the system accuracy for dynamic range during operation.

A test and verification performed in the entire system during development. The validation also performed using KEPCO E&C reference and the results were reasonable of reference. The reference of KEPCO E&C resulted in 100% fidelity thus improved system accuracy as well as plant protection system.

Acknowledgement

This research was supported by the 2021 Research Fund of the KEPCO International Nuclear Graduate School (KINGS), the Republic of Korea.

References

1. World Nuclear Association. Nuclear Power in South Korea. Available on the website https://www.world-nuclear.org/information-library/country-profiles/countries-o-s/south-korea.aspx
2. David Pellerin, Scott Thibault - Practical FPGA Programming in C (2005, Prentice Hall)."
3. Handbook of floating-point arithmetic by Jean-Michel Muller, Nicolas Brisebarre, Florent de Dinechin, Claude-Pierre Jeannerod, Vincent Lef vre, Guillaume Melquiond, Nathalie Revol, Damien Stehl , Serg (z-lib)
4. Numerical Computing with IEEE Floating Point Arithmetic by Michael L. Overton (z-lib.org)
5. Kossiakoff, W. N. Sweet, S. J. Seymour, and S. M. Biemer, Systems Engineering Principles and Practice: Second Edition. 2011
6. IEC, IEC 62566, Nuclear Power Plants - Instrumentation and Control Important to Safety - Development of HDL - Programmed Integrated Circuits for Systems Performing Category A functions. 2012
7. Edition Formal Verification of Floating-Point Hardware Design A Mathematical Approach by David M. Russinoff (z-lib.org) (1)
8. IEEE, IEEE Standard for Software Requirement Specifications - IEEE Std 830-1998, vol. 1998. 1998
9. J. L. Goldman, G. Abraham, and I. Y. Song, "Generating Software Requirements Specification (IEEE - Std . 830 - 1998) document with Use Cases," no. 215, pp. 1-12, 2007
10. USNRC, "10 CFR Appendix A to Part 50—General Design Criteria for Nuclear Power Plants"
11. "IEEE standard 1012." 1998
12. IAEA. Modern Instrumentation and Control for Nuclear Power Plants: A Guidebook. Technical Report Series nº 387. Vienna, 1999
13. B. a Jackson and J. R. Armstrong, "Digital filter design and synthesis using high-level modeling tools," 1999
14. Korea Electric Power Corporation & Korea Hydro & Nuclear Power Co., Ltd. Functional Design Requirements for a Core Protection Calculator System for APR1400. August 2014

15. J. Qian and B. Xu, “Formal verification for C program,” Informatica, vol. 18, no. 2, pp. 289-304, 2007, doi: 10.15388/informatica. 2007.178
16. Design Automation Standards Comittee, IEEE Std 1800.2-2017 : IEEE Standard for Univers Verification Methodology Language Reference Manual., vol. 2011, no. January. 2017
17. Systems Engineering Principles and Practice 2nd
18. USNRC. U.S. EPR Application Documents
19. Korea Hydro Nuclear Power (KHNP) Nuclear Power Education Institute, APR1400 Protection System, training material, pp. 119-120
20. Wang-Kee In; Young-Ho Park; Seung-Yeob Baeg. Development and Assessment of Advanced Reactor Core Protection System. Journal of Power and Energy Systems. Vol 06, n° 02, 2012