HOW TO WRITE DOCUMENT

Document Number	
Issued Date	
Created by	Vu Huynh
Examined by	
Approved by	

How to write document.

Contents

How to write document.

0. Overall.

- The target of this document is to give some advice for how to write an effective document which includes document writing for task/weekly report and design document.
- Furthermore, this document is also suggested some way to develop a paragraph which can be considered a fundamental factor for a design document.
- Reference material:
  - What is “Specification” and how to write it – Dr Keijiro Hayashi.
  - Micro Architecture Specification Guideline.
  - How to write an effective business document – Dung Vuu.

How to write document.

1. Task report.

- In general, the main function of report is to inform the progress and status of a task to leaders, so they can get the information clearly and exactly. Hence, to facilitate the comprehension of readers as well as to minimize the reading time, the criteria for a report are depended on various important factors.
- Firstly, the report format is the most significant point to be taken into account. The format is once defined, it would bring the consistence in the report between members in a project. Thus, any member can apply and follow the same direction. Another point is that the format can help readers to get the information promptly and precisely. Obviously, by applying defined format, readers can skim to get the key points and main ideas as they already knew where the information is located. As a result, the format is recommended to be defined in advance.
- Secondly, the content of report must be “precise” and “concise”.
  - Undoubtedly, “precise” is an important point of the report. The information which is written exactly and clearly can reduce the risk of misunderstanding between writers and readers. Another considered point is to decrease the time for readers to confirm with writer for the meaning or purpose of the report.
  - Regarding the “concise” factor, it is comparable with “precise” in essence. Unlike other document as design document or paper, the report writer must concentrate on how to get information as soon as possible. Therefore, in some cases, a full sentence is not vital in status report whereas the ways to emphasize main idea are higher crucial.
- In summary, writing a report must pay attention to these two elements below.
  - Format must be defined brilliantly.
  - The content must be precise and concise.

1.1. Report format.

- As mentioned above, the format of report must be defined to keep the consistence and the ease in information searching. Depending on different tasks as well as different situation, the report format may vary in styles. The example below introduces one of common format which can be applied for some report.

---

(1). Task A (due day:)

(1.1) Status: <delay/on time/advance/done>

- Delay reason: <optional, no need if no delay>

(1.2) Activity:

(1.3) Issue:

(1.4) Plan/AI:

---

(2). Task B (due day:)

(2.1) Status: <delay/on time/advance/done>

- Delay reason: <optional, no need if no delay>

(2.2) Activity:

(2.3) Issue:

(2.4) Plan/AI:

Figure 1.1.1: Example of report format.

- In general, report needs at least 4 factors as the example, namely Status, Activity, Issue, and Plan/AI. Besides, it is recommended that the report must be clarified by tasks and each tasks need to keep the 4 main factors.

1.2. Report writing – Status.

- Status is the main information to show the overall progress of a task. Thus, the information must be highlighted in the report. To a certain extent, the status can be defined as three types, namely “advance”, “on-time”, and “delay”. In some cases, the status can be used with additional information indicating clearer progress such as “doing - delay” or “done - advance”.
- Furthermore, the significance of status result overwhelms that of other factions such as number or percentage of task in getting the progress. In other words, to evaluate a task, it is vital to take status into account as the number or percentage, in some cases, cannot show the necessary time to accomplish a request. If the number is considered, it means the difficulties of small items in a task are equally evaluated. This kind of consideration is not always true, nevertheless. For instance, a task with 90% done maybe get “delay” status while one with 10% can be regarded as “Advance”. The reason is the difficulties of 10% remained in the former can be exceptional higher than that of 90% in the latter.
- Summarily, the status is seen as a crucial element in the report which needs to focus on at the beginning of reading. In case of project leader, which status types must be remarkably paid attention to.
  - Advance.
  - On-time.
  - Delay.
- In my opinion, the most significant point that needs taking into consideration is “delay” status. Undoubtedly, this status would cause harmful impacts on the schedule of a project while the others are not. From this viewpoint, to minimize the time and satisfy the expectation of project leaders, reporters has to focus on “delay” status writing. As a result, these information is important to report “delay” status.
  - The evaluation of the task should be correctly done for “delay” status such as 3 days delay or 5 days delays.
  - Reporters also need to evaluate the severity of the “delay” status. The table for severity will be shown in later part.
  - Besides, reporters should analyze the reasons and take some actions to overcome the issue by providing solutions for the “delay” status such as resources adjustment, schedule adjustment, technical support, and so on.

Level		Description
0	No impact	No impact on the schedule. At the time of report, the task has not been finished yet, but it can be done in a short of time such as 1h or 4h.
1	Insignificant	The delay causes a small impact on the schedule. However, it can be covered by working OT or spending more time. In general, after taking small actions, there is no delay and the task can be done in time.
2	Minor	Minor delays in achieving objectives, yet the remaining tasks is majority. As a result, in-charge person needs little supports from others to finish the task, or needs to take high actions such as working OT in many days.
3	Moderate	The delay is big and in-charge person cannot achieve the target. Leader needs to take actions to handle the delay such as resources assignments to overcome the delay.
4	Significant	The delay causes high risk to achieve key deliverables, or it can also badly affect continued tasks. Maybe, the schedule must be re-negotiated or the resources for the task must be increased.
5	Major	Failure to achieve one or more key deliverables. It results in negative impacts to customers.

Figure 1.2.1: Level of delay severity.

- The example below illustrates how to write the status in the report.

---

(1). Task A (due day: 20 April)

(1.1) Status: Delay – 3 days – lv2

- Delay reasons:
  - Time for investigating ICC specification is longer (*1) → need technical sharing from ICC designer.(*2)
  - Cost 2 days for training courses (*1) → need to extend the schedule or use weekends to recover.(*2)

Figure 1.2.2: Example of status writing.

- From the example:
  - (*1) shows the reasons of delay, which provides the information for the situation.
  - (*2) provides some solutions or actions for the delay. It would help readers understand and consider how to cope with the delay issue.

1.3. Report writing – Activity.

- The “activity” part would demonstrate the detail activities which are broken down from a task in order to finish it. Furthermore, this part is also used to show which parts has been done or remained in last period.
- In this case, there is no clear format for how to write as “Status” part because the necessary information varies in the kinds of tasks. As a result, project leaders may clarify the kinds of tasks such as TM creation, SVA creation, RTL design, or design document. From the definition, the format for each kinds of task or how to write the “activity” part can be provided.
- Although the format is free from the kinds of task, the focusing point should be negative effects on the sub task as well as main tasks. Similar to “Status”, readers significantly pay attention to harmful impacts, problems, or possible threats which can lead to the fail in the project commitment. As a consequence, the issue must be emphasized in the report with vital information as below.
    - Clearly describe what are the problems or issues.
    - Point out who are responsible for the issues.
    - Provide the writer actions for the issue.

(1.2) Activity:

- Create TM for item2 in PMC checklist:
    - Total: 798.
    - Created: 798.
    - Pass: 681.
    - Fail: 4 (1 TM).
    - Fail reason:
        - 1 TM: wrong value to count EIINT standard function (*1) because Cforest (*2) don’t support to generate interrupt with priority higher than 16 (*1) → wait new Cforest release (ticket #86532) (*3).
- Create TM for item3 in PMC checklist:
    - Total: 708.
    - Created: 708.
    - Pass: 453.
    - Fail: 255 (10 TM).
    - Fail reason:
        - 2 TM: Cforest bug (*2) when counting in guest mode (*1) (ticket #86445) → wait new Cforest release.(*3)
        - 4 TM: Interrupt BGFEINT, BGEIINT, GMEIINT, and GMFEINT function have not been supported (*1) in Cforest (*2) yet (ticket #81585-note2) → wait new Cforest release.(*3)
        - 4 TM: limitation of Cforest (*2) (ticket #86442) - PINC is unsupported (*1) → wait RTL simulation environment to confirm.(*3)

Figure 1.3.1: Example of activity writing for TM creation task.

- From the example:
    - (*1) indicates the fail description or fail detail after analyzing.
    - (*2) shows the responsible side or the root causes of the fail.
    - (*3) illustrates the action which must be taken for the fail.

- Note:
  - o This kind of writing may be chaotic in the report which may lead to misunderstanding or difficulties to identify the content of the issue. Hence, the format below brings more convenience for readers.

TM No	Fail reason.	Responsible.	Action.	Control ticket.
1	Wrong value to count EIINT standard function in case priority is over 16.	Cforest	Wait new Cforest release.	#86532
4	PINC is unsupported because of Cforest limitation.	Cforest	Wait RTL simulation environment.	#86442

  - o In general, the issue must be reported to in-charge side/person and the project is mainly controlled by Redmine system. Thus, the Redmine ticket number is required in the report for others to refer detailing information.

1.4. Report writing – Issue.

- "Issue" is another factor which must be carefully considered in the report. It is obvious that people may fail to accomplish their project objectives by "Issue". Consequently, this part is as important as "Status" in the report, so writers should precisely describe what are the issues needing dealing with.
- Another discussed point is the way to clarify what should be regarded as "Issue". To a certain extent, what can pose threats to project achievement should be treated as "Issue", including risks in the present or possible ones in the future. From the perspective, these cases below can be written in "Issue" part.
  - Firstly, the problems with which members struggle to cope should be considered. In some cases, although the trouble can be solved, the time consumption is enormous. Obviously, one of criteria for a project commitment is schedule, so the consideration for time should not be reckless.
  - Secondly, some unclear inputs or detailing specifications that cannot be totally understand should be written in "Issue" part. Absolutely, the vague ideas can result in the divergences from original purposes, which leads to the failure in the development or the verification activities due to the misunderstanding. As a consequence, ambiguous information can be treated as possible "Issue" harmfully affecting the success of a project.
  - Finally, one of consideration points is potential risks to the projects or tasks. It is undoubted that people may find it difficult to identify or predict the dangers in the future. However, the proper considerations for likely threats are definitely appreciated.
- With regard to how to write an issue, the first noticeable element is how to describe the issue precisely and concisely in order to facilitate the comprehension of readers. Additionally, what people need to overcome the issue should be mentioned as well. Hence, readers can get the overall situation and essential actions for an issue.

(1.3) Issue:

- Cost a lot of time to investigate "Instruction Cache" specification (*1) → need technical sharing from designer to fasten verification phase.(*2)
- The function of SVLOCK in host mode is unclear in RH850v2 specification (*1) → need REL confirm in ticket #86414 by 18th Aug to start checklist creation.(*2)
- The necessity of MPU IFV environment to finish MPU verification by 15th Aug (*1) → need supports to create IFV environment by 1st Aug.(*2)

Figure 1.4.1: Example of "Issue" writing.

- From the example:
  - (*1) demonstrates the situation of the issue.
  - (*2) shows the request or the requirement to handle the issue.

1.5. Report writing - Plan

- Although this part is the least important part in the report, its significance cannot be under evaluated. The “Plan” provides information for tasks or jobs in the future, so it should consist of some requirements as below.
  - The task information or activity must be clearly written with specific due day.
  - The due day must be reasonable in considering the volume of tasks as well as the ability of in-charge person. In other words, the estimate must be as correct as possible, so the leaders can exactly evaluate the status of the whole project in the future.
- One of recommendation for task estimation is to break the task down into small tasks; then small tasks would be considered in details to get correct deadline. The smaller tasks are divided, the more correct the estimate is.
- Another point is how to take experience for the prediction of each tasks, so similar tasks can be forecasted more precisely.
- Besides, in some cases, people should think for the possible risks in the future. This is the likely happen cases which can cause badly impacts on the project. Hence, leader or other members can consider the back-up plan if the situation is needed. Similar to “status”, writers should put the estimated severity for the risks.

(1.4) Plan:

- Task in next week:
  - Create 238 patterns for MPU AVP (*1) by 18 Aug.(*2)
  - Run SVA and TM with RTL environment (*1) by 20 Aug. (*2)
  - Update SVA/TM based on checklist feedback from REL (*1) by 22 Aug. (*2)
- Risk management:
  - License may be critical in next week → Run SVA and TM can be 2 days delay from the schedule, yet the running tasks can be easier on weekend. – lv1

Figure 1.5.1: Example of “Plan” writing.

- From the example:
  - (*1) give information about the detailing activity of the tasks.
  - (*2) shows the deadline for each tasks.
  - People, sometimes, should think and forecast the possible risks in the future.

1.6. Report writing – Practice.

- The example below shows a report with various mistakes in comparison with listed points in the document. What should be done here is analyzing and correcting the mistakes in the report.
- For the reference answer, please refer Appendix Report correction.

---------- **(1). Interrupt verification for RH850v2 (due day: Middle Jun)** ----------
(1.1) Status: On-going – 90%
(1.2) Activity: verify FEINT and DBINT interrupts.
- Total: 419
- Created: 400
- Pass: 320
- Fail: 80 (7 TM)
- Fail reason:
o Related to PINC (5 TM): wait the available of RTL simulation.
o STM.GSR and LDM.GSR function in Cforest has bug (2 TM).
(1.3) Issue:
- Spend time confirming old items with HVCFG.HVE=0, which costs many times due to many difficulties in fail analysis.
- Need sharing about INTC1 function.
(1.4) Plan:
- Continue create remaining TM → plan to finish by June 28.
- Confirm the fail TM again with new Cforest.

2. Design document.

a. What is a design document?

- Design document would show the specification which is used to explain a detailed instruction for how a microprocessor or IP are made.
- Furthermore, design document, sometimes, explain the reasons for the design such as “why this implementation is chosen”, “what are the benefits of the implementation”, or “what are the problems of the implementation to not follow the methods”. Therefore, readers can completely understand the designers’ purposes and ideas in which are applied.

b. What are the main functions of design document?

- Firstly, design document, which demonstrates the thinking or understanding of designers, can be used to confirm the equivalences with input specification from customers. To a certain extent, designers fail to completely understand the requirement or expectation of customers, so design document solves the issue of possible mismatches between customer’s expectation and designer’s understanding.
- Secondly, for verification process, the availability of design document plays a crucial role in the accomplishment of the qualities. In case verification leaders or members can understand the design document totally, it would help:
  - Avoid the lack of verification items due to the absence of design’s acquaintances.
  - Decide an effective way to verify each function.
- Regarding design phase, the design document illustrates the opinions, ideas, or implementation methods of designers, so by checking the document, the issue may be found at an early phase. Hence, the cost for design modification can be declined.
- Additionally, when a new designer reuses the old design as a legacy, people find it essential to firstly comprehend the design document to avoid mistakes in modification or application.

c. What are harmful impacts on a project with no design document?

- During the phase of input specification investigation, no designers can provide a concrete evidence for their full and correct understanding the requirements in the specification. The lacks of design document may lead to the dangers of wrong understanding or the omissions of specification expectation. Consequently, the risks would result in the customer dissatisfactions with the outcome of designers.
- Similar to the input specification, the insufficient verification items may come from the lacks of design document, for verification members cannot judge whether the verification items are enough or not. Thus, the potential bugs may come from missing descriptions.
- In some cases, the designers who modify the logic or update the design are not the original designers. Hence, if the document is not clear enough to describe necessary information for others to get the ideas, or reasons of implementation, the risks to cause design defects appear.

d. What are the requirements of a design document?

- Two significant factors required in a design document are “Quality” and “Quantity”. Definitely, the absence of these two key points can lead to a possibility of misunderstanding between different sides such as customers – designers or designers - verifiers, or mistakes in logic design.
- Considering the “Quantity” of document:
  - Obviously, the information that needs for the logic implementation must be illustrated in design document. Thus, the designers can develop the logic implementation sufficiently. It is obvious that the descriptions or conditions in design document closely connect with detail implementations, so the lacks of conditions in the document may lead to the defects in considering the implementation in this case.
  - Moreover, essential information should be written into one document only. In some designs, the implementations or conditions are separately described in different documents. Hence, it is hard for others to refer the information in various documents. For instance, a new designer who is not original one can refer documents to modify the logic

implementation, yet the new one may forget to check some documents, which can cause implementation mistakes. Consequently, the possibility of lacking considerations, which can bring errors or bugs in logic implementation, is said to be the results of unwisely divisions for documents.

- Furthermore, the excessive information causes more harm than good in some cases. Complicated descriptions in the design document may eclipse the main or essential points in the design. Hence, people may perceive it difficult to investigate or realize the important information for their design modification task or verification making one.
- Summarily, the “Quality” feature requires “sufficient descriptions”, “putting in one document”, and “avoiding excessive information”.

- As regards the “Quality”:
  - The first criteria for “Quality” is the consistency between design document and logic implementation. In many cases, a designer develops a method for logic implementation, but the real implementation seems incorrect or causes problems. As a result, the modification is needed to fix the issue, so the updating design document to match with logic implementation is necessary as well.
  - Another point is the design intent explanation. In other words, the ideas of original designers have to be precisely explain in the design document, so other people such as reviewers, or verification members can understand the reason of the implementation. For example, they can get “why this implementation is chosen”, or “what are the good points to develop the design following this methods”. (Real example is described in Appendix Disappearance of design result)
  - Additionally, the description in the design document must be clear for all readers to get the same understanding when reading the document. Undoubtedly, if a writing style can result in several comprehensions, the risks of creating bugs due to the misunderstanding may happen. Therefore, it is recommended that the document must be rewritten whenever various interpretations are available in the document. (Real example is described in Appendix Ambiguous description)
  - Lastly, in some implementations, the opposite conditions should be mentioned such as the explanation for “what would happen if the condition is false”. Although the workload to make a design document with negative conditions is extremely colossal, it is worth to process.
    - The first benefit is to help verification members to make a full checklist in both positive and negative conditions. In many cases, bugs locate in the negative conditions which are missed in both design and verification. For example, condition A leads to operation B, so the question is what would happen if condition A is unsatisfied? By describing the negative operation in design document, verification members must create items to verify the cases so the verification activities are assured of increasing qualities. (Real example is described in Appendix Negative condition)
    - The second one is to clearly understand the description, so readers can get full the idea of designers in both positive and negative ways. In certain circumstances, readers or reusing designer can totally know why the implementation is chosen or what the best solution is in this case.

- As a result, how to write documents and what should be put into documents are stressful issues of designers or document creators. To a certain extent, when making document, it is essential to consider the following questions to decide how to write documents:
  - What are the functions of the design?
  - What should be written in the design document?
  - What are the purposes of the implementation?
  - Why are the implementations chosen?
  - What do happen in negative conditions?

2.1. Design document – Partition.

- Format is the most important things needing considering at the first phase of a project as a good format can bring a well-organized document. The advantages from a good format are various.
  - The format, to a certain extent, provides fundamental requirements for the contents of a document. It would help designers, especially new designers to achieve the foundation of document making.
  - The format contributes plus points to preventing the messy development of design document, which may lessen the prospects of logic implementation issues as well as verification ones caused by complicated or ambiguous description.
  - The format also reduces problems of various writing styles. Assuming that a project has many designers, in case no design format is defined, the writing styles or the ways design documents are developed vary in designers. As a result, it is hard and time consuming for readers to get information from different design documents.
- Overall, the format must be defined in advance to better the document development phase. Obviously, the format would depend on different projects or different design leaders, yet the availability cannot be denied.
- From the viewpoint of essential format, how to define an effective format poses several challenges for project leaders or design leaders. The following demonstration suggests one of the way to make the design document and the design document format.
- Normally, to make a design document, the process would be from the top level to lower level, or from general idea to detail implementation. Therefore, to start a design document, the beginning point must be the input specification which is the fundamental principle to develop the design.
- Design document can be split into 3 main parts:
  - Functional specification.
  - Interface specification.
  - Implementation specification.

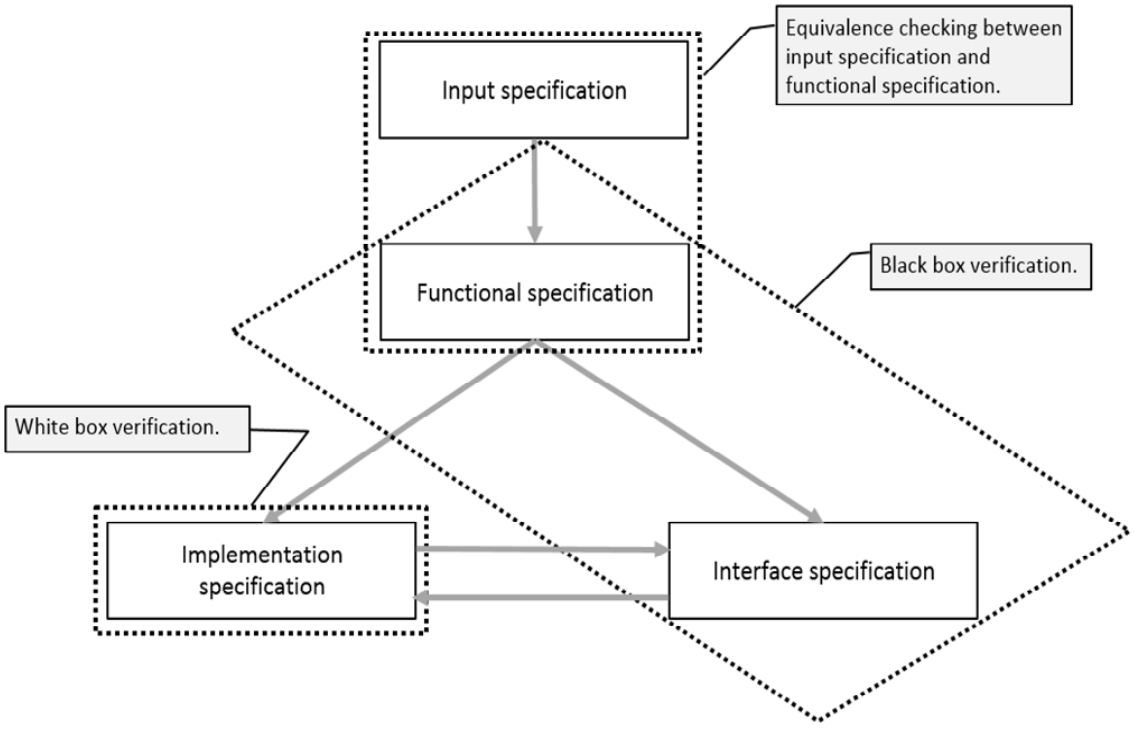


Figure 2.1.1: Design document partition.

- The input specification in which the expectations or requirements of customers are located is the foundation to make the functional specification. In general, to create a functional specification, the simplest way is to list up all the necessary functions which are needed to satisfy the requests in input specification. The functions can be divided into different levels from general ones to detailing ones (refer figure 2.1.2). Moreover, the main purposes of functional specification are:
    - The foundation to develop interface specification and implementation specification.
    - Black box verification application.
    - Sufficiency checking between input requests and supported functions.

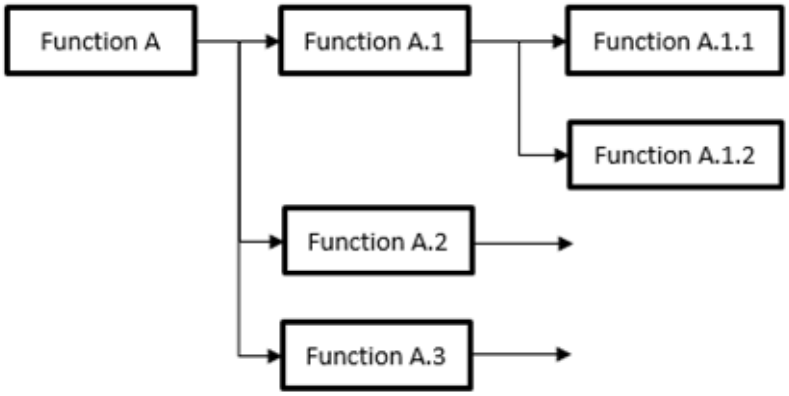


Figure 2.1.2: Example of functional specification creation.

- Once the functional specification is made, the recommendation is to start with the interface specification. It is a little complicated to consider the interface specification without implementation document, yet the former can be updated when developing the latter. Overall, from the functional specification, designers can imagine what they need to develop the functions. Hence, the implementation specification can be created based on that consideration. Besides, the output specification would illustrate what information is generated by the functions to subsequent units or modules. The primary functions of interface specification are used to.
    - Apply in black box verification associated with functional specification.
    - Provide protocol between connected units or modules.
- Similar to interface specification, implementation document is made based on functional document. In addition, the reference to interface specification is also essential as the interface of a unit would affect the implementation methods. Hence, in case the implementation document needs supporting with more interfaces or special constraints to implement a function, the feedbacks to interface one must be done and get the approval from related designers. Additionally, the purposes of implementation document are.
    - Confirm the implementation methods in the early phase.
    - Use in white box verification.
    - Facilitate the legacy for later development.

- Moreover, a simplest way to make the implementation specification is to apply functional specification as a foundation input. To be more precise, the implementation document would associate with functional document by each items. For instance, in functional specification, the functions A is listed up, so the implementation document must describe how to develop the function A. As a result, each functions in functional specification would correspond with each implementation items in implementation one. From this making, the equivalence between functional specification and implementation specification can be checked. Hence, the possibility of lacking items may be eliminated.

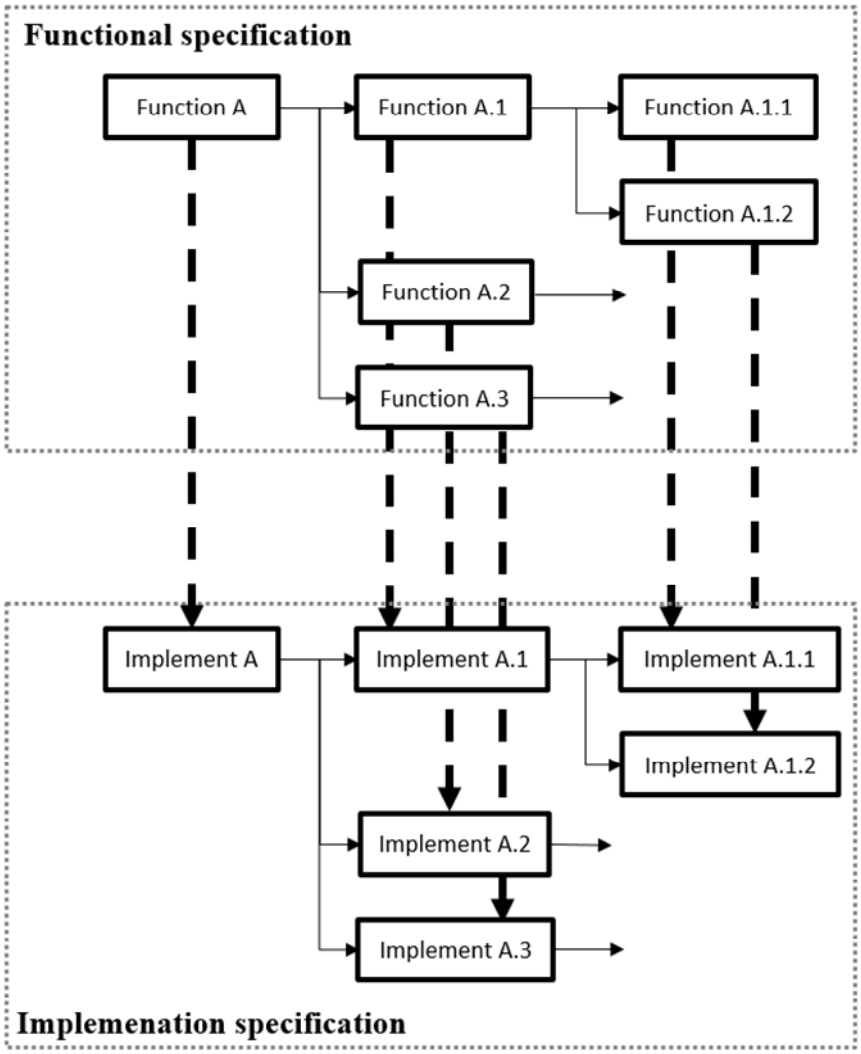


Figure 2.1.3: Functional specification and Implementation specification mapping.

- In conclusion, a design document must consist of 3 main parts which are functional document, interface specification, as well as implementation specification. From the following parts, the references and the recommendations for each specifications are described.

2.2. Design document – Functional specification.

- As mentioned above, the input to make functional specification is the input specification in which customer requests or written standards such as IEEE standards are defined. For various designers, the functional specification writing, sometimes, is similar to input specification, yet it is not totally accurate to a certain extent. The functional specification demonstrates abstract concepts of designers for how to satisfy customer requests.

* Note: one of important points to develop a functional document is avoiding the detail logic implementations description which should be located in implementation specification.

- These items below are the recommended information which needs to be written in functional document. To a certain extent, they can be regarded as functional document format.

  - Functions introduction and description: the overall ideas to adapt customer requests are written in this part.
  - Function list: the function list would show all necessary functions in a table in level from general functions to detailing functions.
  - Function partition: based on the function list, the function would be divided into functional block or group. Then, the block diagram, which is the foundation for logic implementation, is created.
  - Function interaction: this part would give information about how these functions connect with each other or how they affect the operation of others. From this part, the matrix for functional combinations or flow chart of processes are generated, which helps to improve both design activities and the qualities of verification-item making.

- Various concerns are hold on the reasons of why the functional specification is crucial. Actually, the plus points of functional specification making are remarkable.

  - The functional document shows the background of designers' understanding for the requirements or expectations of customers. Thus, the possible threats of insufficiency between customers and designers may be identified at the early stage by confirming functional specification, which reduces the workload, time, human resources, and further additional costs needed to deal with the problems.
  - Furthermore, regarding the legacy in design, the functional specification plays an important role for other designers to comprehend the overall ideas of original designers or the general concepts of the design.
  - The functional specification helps to organize the structure of a design. By developing the document, the functional blocks are created, so designers and reviewers can check the partitions of functional blocks whether the division is effective or not. Therefore, the modifications for the design organization can be improved in advance, which betters the logic implementations in design phase.
  - The omissions of cases in functional verification may be avoided.
  - The complex cases which combine various functions may be generated from the functional document. Thus, the verification qualities can be increased.

2.2.1.Function description.

- The function description is a part to demonstrate specific functions for how to support the requirements from input specification.
- The reasons for function description implementation are:
  - From function description, people can comprehend what are vital points to satisfy the requests of customers, and customers are also able to check or confirm whether their expectations can be accomplished with the implementation or not.
  - In addition, from the function description, designers can consider how to support requirements from information of function description. Then, the function list and function partition would be generated based on the information. Thus, the organization of the design may be improved from the well-ordered developments of function design.
  - Function description is the essential entrance into design details as it describes the overviews or main supported functions in which the design must be satisfied. Other designers can get the top-view of the design to process the following tasks such as functional verifications or further design investigations.
- The example below illustrates how to write general descriptions in functional document.

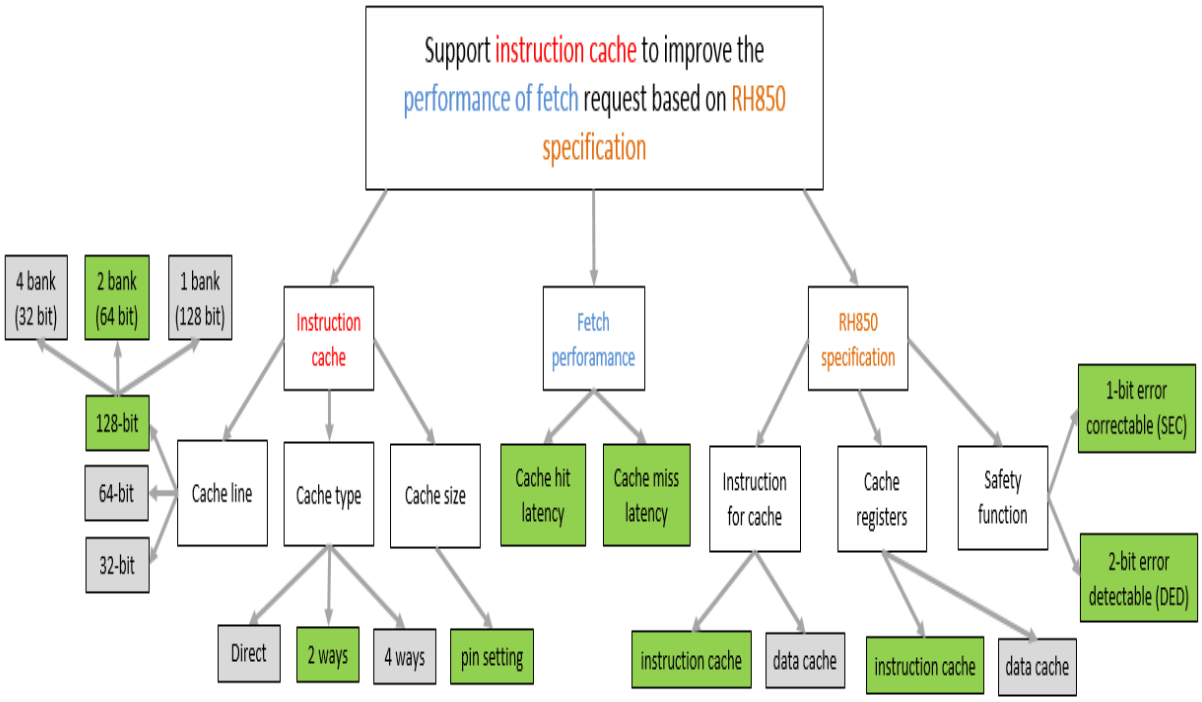


Figure 2.2.1: sample mind map to create function description.

- From the mind map, designers can create function description as below.

Input specification.	Function description.
Support instruction cache to improve the performance of fetch request based on RH850 specification.	- Support instruction cache with 128-bit data fetch request from cacheable areas. The fetch latency for cache hit is 1 cycle, otherwise the latency depends on memory latency. - Support instruction cache with 2-way associate access. - The size of instruction cache can be configured by pin setting for 0KB (instruction cache is deleted), 4KB, and 8KB. - Instruction-cache data are 128 bits which are split into 2 banks with 64 bits. - Instruction cache supports instruction-cache instructions which follow the specification of RH850 architecture (data-cache instructions are unsupported). - Instruction cache supports instruction-cache system registers following the specification of RH850 architecture. - Safety function: single bit error correctable for data, and double bit error detection for data.

* Note: As written above, the function description just introduces the overviews or main functions of the design, and designers should not put so much detail implementations in this part.

2.2.2.Function list.

- Function list is the list which illustrates functions of a design. The list is made from top to down for functions in the design, so the input of function list is function description. To be more details, the function description can be regarded as the first class functions of the design. From that first classes, second classes are clarified to explain how the first class is supported. Then, the following classes are applied the same. From this way, n class is created, and the number of class which needs describing depends on the complex of the designs. For usual designs, two or three classes are enough to put into function list.
- The example below shows the sample of function list.

First class	Second class	Third class
Function A	Function A.1	Function A.1.1
		Function A.1.2
	Function A.2	Function A.2.1
		Function A.2.2
		Function A.2.3
	Function A.3	-
Function B	Function B.1	Function B.1.1
Function C	Function C.1	Function C.1.1

-What are the intents of developing function list?

- o The first reason for function list creation is that it is the foundation of the design. Once the function list is available, designers can continue developing following design steps as function partition. For example, from the function list, designers can decide that A.1 and B.1 should be implemented in block_1. It brings various benefits for how to well organize the design at first, and how to produce an effective design in later.
- o The combinations between functions in the design can be extracted from function list, which facilitates how to decide implementation methods to support these complex functions.
- o Furthermore, to satisfy the specification requests from clients or written standard, various approaches can be taken into developments. Hence, the function list implementation can partly show the intents of designer such as why this mean is applied or why it is necessary.

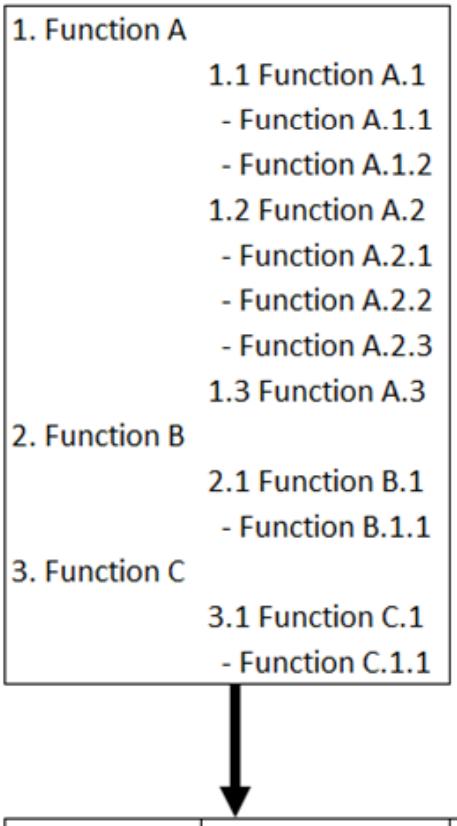
- What are the benefits of function list?

- o To enable verification engineers to sufficiently verify the functions in the design, which would meet the requirements of functional verification. Besides, the possible lacking verification items can be detected by checking the equivalences between function list and items in functional verification checklist.
- o Other designers can understand not only general functions of the design but also the detailing ones, so the verification and design legacy can be developed easily.
- o From function list, the complicated cases which merge between many functions in the design can be created from the function list (function interaction – be described later). Therefore, the consideration for design and verification activities can be better.

- The example of function list.

First class	Second class	Third class
Support instruction cache with 128-bit data fetch request from cacheable areas.	Detection logic to clarify whether an access is cacheable area.	Based on the address access and request valid, a decode logic is implemented to check cacheable area or not. Then, generate 2 signals which indicate cacheable area and non-cacheable area.
	FSM to control the operation in ICC when a fetch is valid.	Enable cache read request to get data for hit/miss judgment when a fetch is valid.
		Switch the operation to issue fetch request to ROM area to get data or use data from cache when hit/miss judgment result is valid.
		Control the fetch request when busy from memory is responded. FSM must hold the request and wait until memory is ready.
		Control the response phase when data from memory are valid. Inform data valid to other units to take data to output to CPU and to update into cache memory in case of cacheable area.
	Hit/Miss judgment.	Based on data from cache, check whether a request is hit or not. Then, inform the result to FSM for further operation. Because of 2-way associate cache, the comparison must be doubled to support 2 way checking.
		Detect multiple hit error when 2 ways are both hit. Inform the result to FSM for operation and CPU to generate exception.
	Cache access interface.	Support read/write interface with cache memory. Because of 2 way associate cache, there are 2 ports to access to 2 ways of cache memory.
		Read data from cache for hit/miss judgment.
		Write data to cache when data from cacheable area are valid. Way selection would be developed based on LRU algorithm.
	Cache buffer is implemented to reduced access times to cache (power consumption optimization).	One line buffer is implemented to keep newest data from cache for next time request. In case the request matches with buffer, the data would be taken from buffer and cache memory would not be activated to reduce the power consumption of memory access.
		A comparison is implemented to check whether the access is hit with buffer or not. Output hit result to cache interface to cancel cache activation when hit.
		Control logic to handle buffer updating when data are valid.

- In some complex designs, designers find it hard to implement the function specification following the table style, so tree function can be a better choice instead of function list to describe the details of functions.
- Tree function can be created from a specific function and break down it into sub-functions as the function list. Then, designers can continue to process the explanation of each functions from top to down and from general to specific. The tree function can be seen from below example, but the function list (MAIN IDEAS ONLY) should be created for later uses such as function partition or function matrix creation.



First class	Second class	Third class
Function A	Function A.1	Function A.1.1
		Function A.1.2
	Function A.2	Function A.2.1
		Function A.2.2
		Function A.2.3
	Function A.3	-
Function B	Function B.1	Function B.1.1
Function C	Function C.1	Function C.1.1

Figure 2.2.1: Example of tree function and function list.

2.2.3.Function interaction.

- All functions described in function list are not independent with each other's, so some may have the relationship. Hence, during the process of function specification making, how the function interacts with others needs considering and the explanations for them are essential.
- The function interaction can be generated by referring the function list result. There are various kinds or formats for function interaction creation, yet matrix table and flow chart are the common ones. The examples below show the matrix table and flow chart.

	Fetch ROM	Fetch CACHE	Hit/Miss judgement	Address generation	Response error	Response data selection
Fetch ROM		x	o	o	x	x
Fetch CACHE			o	o	x	x
Hit/Miss judgement				o	x	x
Address generation					x	x
Response error						o
Response data selection						

o: these functions have relationships together.
x: there is no relationship between functions.

Figure 2.2.3: Example of matrix table.

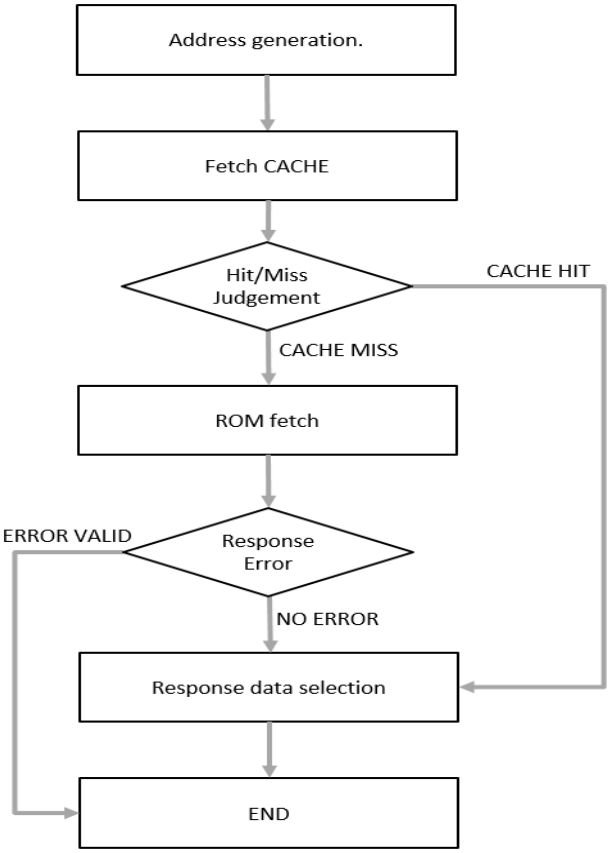


Figure 2.2.4: Example of flow chart.

- The intents of function interaction in design document are:
  - The interaction shows the relationship between functions in the design. Sometimes, it can indicate why the combination is available or unavailable.
  - If the combination is not clearly shown, unnecessary logic may be implemented to handle unreachable combinations. As a result, the logic implementation becomes complexed.
  - In case function partition is done without function interaction, unappropriated considerations for function partition result in huge modifications for the design. Example: because A1.2 and B1.1 have the close relationship, they should be located in a same sub-module.
  - The interface specification between sub-modules can be precisely created as the relationships are clear from function interaction table. A sub-module can recognize functional relations with others, so proper interface signal or specification can be generated.
  - From the interaction table, the possible combination for test pattern creation can be reduced by removing unreachable conditions.
- The risks without function interaction design:
  - If the functional verification is generated based on function list only, the possible lacks of complex combinations between functions may lead to insufficient verification.
  - Other engineers can re-use and understand the relationship between functions in the design for further development or verification.
  - For some complicated connection between functions, the designers may lack or even forget the considerations for that item, if they are not well written in design document.

2.2.4.Function partition.

- The function partition created from the function list results in physical blocks. In other words, from the function list, designers would organize their design. For example, the submodule_1 should implement the function A.1 and function A.2, the submodule_2 should consist of A.3, B.1 and C.1.
- In ideal case, the result of function partition would be matched with the function list. Nevertheless, in some cases, because of design intents or special purposes, a function in function list can be divided into different submodules such as function A in the previous example. As a result, the requirement for function partition is all functions in function list must be located in submodules after function partition is done.

Memory protection function.	Load/Store violation checking.	Load/Store access type detection.	MdpDetect module
		Address comparison with setting registers.	
		System protection ID checking.	
		Permission checking.	
		Error generation.	
	Self-check function.	Address comparison with setting registers.	
		Permission checking.	
		Information output for system register updating.	
	System registers.	Read/Write system registers.	SysReg module
		Setting register information output	
		Self-check function information updating.	
	Fetch violation checking.	Address comparison with setting registers.	MipDetect module
		System protection ID checking.	
		Permission checking.	
		Error generation.	

Figure 2.2.2: Example of function partition.

- The main reasons of function partition are:
  - To confirm the equivalence between function list and functions of submodules. In case there are some differences between them, the consideration to judge why the mismatches occur must be taken. Thus, designers can confirm whether it is a lack of functions in function list or functions of submodules, or these functions are redundant and can be removed.
  - To create the block diagram of the design from function partition's result. Obviously, the block diagram of the design can easily generate from the function partition, and the relation between submodules can be seen from the function partition.
  - To process the consideration for whether block divisions are sufficient or not at an early phase, so the modification if need can be done sooner.
  - To enhance the understanding of other designers for the relationship between submodules in a design.
- The problems for the lacks of function partition are:
  - Other designers and reviewers cannot consider the effectiveness of function partition result, which may lead to insufficient physical block partition of the original designer. As a consequence, inappropriate implementations for physical block partition result in enormous re-design tasks in the following processes.
  - Other designers find it difficult to understand the relationships between submodules, which costs resources for design legacy and verification.
  - Mistakes may occur during the design phase as designers lack the implementation for some functions in submodules.

2.3. Design document – Interface specification.

- In interface specification, they are split into 2 kinds of interface documents, namely external interface and internal interface. The former indicates the connections and the relationships between modules while the latter shows the connections between sub-modules in the design.
- All of these interface document have the same strategy to be developed by referring the functional specification (focus on function partition and function interaction). In this document, it just shows the way to make interface specification in general as the strategy is similar for both internal and external interface.

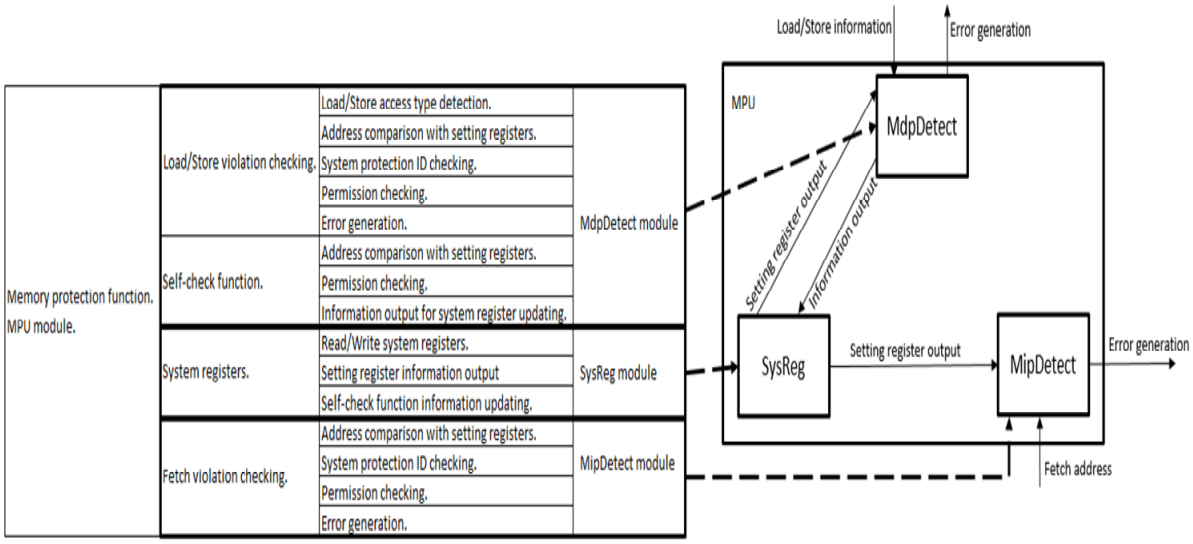


Figure 2.3.1: interface specification development from function partition.

- In interface specification, the content requires 2 principle parts:
  - Interface list which describes signal, bit width, reset value, and short description.
  - Interface relationship and explanation which show the connection between signals, the meaning, or the protocol.
- The intents of interface specification:
  - Show the protocol between modules, which includes expectations, or requirements between modules to handle functions or communicate in the design.
  - Show the relationship between signals in the design such as control signal and data signal.
  - Confirm the necessary inputs to support each functions in the design.
- The risks of lacking interface specification:
  - Mismatch in the operation between modules, especially hand-shake signal.
  - Mismatch in the expectation/time for when the signal is valid or when the data should be available.
  - Lack necessary signals to support a function between modules, which results in the high workload to re-design the RTL code as well as design document.

2.3.1. Interface list.

- Simply, the interface list is a table which shows the input or output signals of a module with the description to explain the characteristic of those signal. The format and the content of interface list depend on each designers or specific formats of the project. However, the information as the example below is the principle content when developing interface list.
- The example below shows the fetch interface between ICC and ROM memory. This is one of the functions which are supported in ICC. The interface list would be divided into groups which is implemented to support for a specific function in the design.

Fetch interface between ICC and ROM memory.			
Signal	**In/Out**	**Initial**	**Description**
RomAccReq	Out	1'b0	Request access to ROM area from ICC. - 1'b1: ICC request fetch to ROM. - 1'b0: ICC does not request fetch to ROM.
RomAccAdr[31:0]	Out	Undefined	Address of access from ICC to ROM area
RomAccRdy	In	1'b0	The status of ROM memory. - 1'b1: ROM memory is ready to receive access. - 1'b0: ROM memory is not ready to receive access.
RomResVld	In	1'b0	ROM data are valid from ROM memory.
RomResDat[127:0]	In	D.C	ROM data are responded from ROM memory.
RomResRdy	Out	1'b0	ICC status to receive ROM data. - 1'b1: ICC is ready to receive data from ROM. - 1'b0: ICC is not ready to receive data from ROM.
RomCfgSize[2:0]	In	Optional	ROM memory size: - 3'b000: 128KB - 3'b001: 256KB - 3'b010: 512KB - 3'b011: 1024KB - Other: unsupported - setting is not recommended.

- The "Signal" describes the interface name with the bit width. In some formats, designers create additional column to show the number of bit for each interface. Nevertheless, some signals declare as A[31:4], so the number of bit such as "28 bits" cannot fully illustrate the characteristic of the interface while the example above can show it clearly. In addition, for this example, data signals are indented by some spaces and data signals are written right after their control signals. This rule is also a kind of personal styles of each designers and it is not mandatory.
- The second column is "In/Out" which is used to describe the signal is input or output. It is recommended that input and output signals should be arranged by group of functions. For example: 3 first signals can be regarded as "request interface from ICC to ROM", and 3 next signals can be seen as "response data interface from ROM to ICC".
- The "Initial" column is one of the most important parts in interface list. Although the contents are quite simple, the meaning of each part is significant. Overall, this column shows the initial value of each signals or the value after reset released. The meaning of "Initial" can be categorized as:
    - Exact value (1'b0, 1'b1, or any defined value): it means after reset released, the system expects correct that value. The value cannot be set randomly, which can cause defects in the current design. Normally, this definition is used for control signals.

- Undefined: this is specified for output signal, and usually uses for data signals which need control signals to valid the value on that interface. In this case, the current module would not control the value of output via this interface. Other modules should not use the value if there is no activation from control signals.
- D.C: it is abbreviated form of “Don’t care” which normally uses for input data signals. It means after reset release, the signal can be any value. The operation of current module would not be affected by the value of this signal if there is no corresponding control signals activating the input data signals.
- Optional: it is usually used for input signals which are pin configuration signals. The operation of current module would be affected by the input value of the signals. With various value inputted, the operation of module would be different. Taking the “RomCfgSize[2:0]” as the example, this signal would inform to ICC the current supported ROM size in the system. Thus, ICC would control the access of itself based on that value to avoid out-order-supported area accesses. In general, it is recommended to set this signal (the value is valid) before reset released, and keep stable during the operation of system.

- The final column is used to describe the meaning of current signal or show the status/meaning based on different value. The description should be precise and concise to help other readers get the idea easily.

2.3.2. Interface explanation.

- This part would describe the relationship between interface and the protocol between modules. The information in this part is really important to confirm or judge the sufficiency of timing implementation between modules. For instance, when will the data become valid or which signals valid the data are illustrated in this part, so other modules can confirm or refer to implement their logic.
- The plus points are:
  - From interface specification, each modules would consider the sufficient implementations for suitable data flow. The timing or performance implementation can be checked sooner, so appropriate modifications can be done at the early phase.
  - The explanation can be applied in interface verification activities which strongly relate to the relationships and the protocol of modules in the design. Hence, sufficient verifications can be taken based on the specification.
- The example below illustrates how to write interface explanation or what should be contents in it. In this example, the fetch interface between ICC and ROM memory is taken to follow the stream from Interface list part. If the example for interface explanation is written by "*Italic style*", while "normal style" is used for writer's comments or explanations.

Fetch interface explanation between ICC and ROM memory.

- *RomAccReq: this signal indicate the request from ICC to ROM memory to fetch data. When the signal is 1'b1, the request from ICC to ROM is valid, so it is the control signal to validate the fetch packet to ROM. The activating type is high level detections.*
- *RomAccAdr[31:0]: this is the ROM address which needs to read data. The address is valid only when RomAccReq signal is 1'b1. Otherwise, ICC would not control the value of the address. Although the fetch address is 32 bit, 4 last significant bits (RomAccAdr[3:0] == 4'b0) are fixed to zero as ICC requests 128-bit data fetching.*
- *RomAccRdy: this signal indicates that ROM is ready to receive the request from ICC. ICC only care the value of this signal when RomAccReq is 1'b1, otherwise the value can be zero or one.*

- The explanation above is used to explain the relationships between fetch request interface of ICC and ROM area. In general, the status of control signals should be shown to indicate when they are valid. Besides, the data signal must point out which is its control signal.
- The explanation just explains the meanings and relationships of fetch request part, which is not enough to show the hand-shake protocol of ICC and ROM memory. Hence, additional descriptions or constraints must be added to illustrate the protocol. This part is really significant in view point of communication between modules (ICC and ROM in this example).

- *Fetch request protocol between ICC and ROM memory:*
  - *ROM memory can serve one request at one time only, so the new request must be hold/waited until the previous request have finished the transfer (data are responded and taken from ICC).*
  - *During the time of pending request (RomAccReq == 1'b1 and RomAccRdy == 1'b0), ROM memory expects that the request valid (RomAccReq) and the address request (RomAccAdr) would be kept stable in next cycle. The request signal can be negated only when RomAccRdy is 1'b1 (the request is received from ROM memory). After RomAccRdy is 1'b1, RomAccReq should be fall in next cycle to keep the single access requirement from ROM memory.*
  - *When the request from ICC is pending, ROM memory can respond ready signal at any time. There is no constraint for the number of pending cycles in ICC.*
  - *The ready signal from ROM memory (RomAccRdy) has its meaning only when there is a request available. Otherwise, ICC would not care what the value of the signal is.*

- The explanation for the protocol or constraint between modules should be included and be carefully written. The missing information may lead to the communication problems between modules, so the huge work to redesign is generated.
- To clearly illustrate the protocol, some timing charts can be added for other people to easily get the requirements. For example:

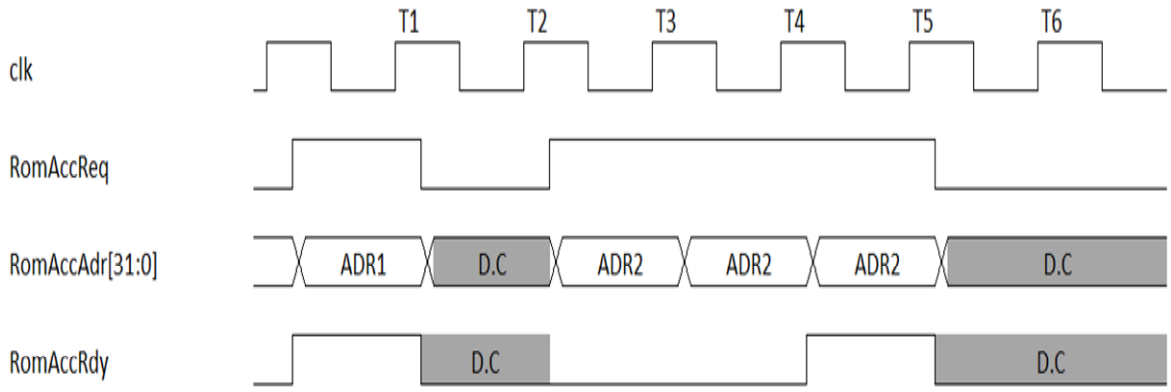


Figure <number>: the example for ICC and ROM memory protocol.

- *T1: RomAccReq issue a request to read data at RomAccAdr == ADR1, and RomAccRdy indicates that ROM memory can receive the request immediately. This is an example to show the best case of request (no pending).*
- *T2: when the request is accepted, RomAccReq signal would be de-asserted in this cycle. When RomAccReq is 1'b0, the value of address (RomAccAdr) and ready (RomAccRdy) is "don't care", so they can be any value. At this cycle, if the data of ADR1 are not valid, ICC cannot issue new request to ROM memory to follow the constraint of single access in ROM memory.*
- *From T3 to T4, the RomAccReq is valid, yet ROM memory is busy. Hence, RomAccRdy is 1'b0 to show the status of ROM memory. During the pending phase, the request and the address from ICC must be kept stable as the requirement of protocol.*
- *T5: RomAccRdy is asserted to indicate the availability to receive request from ICC.*
- *T6: RomAccReq is cleared, and the operation is same as T2.*

- Note:
  - The explanation for the timing chart is important for others to fully understand what is described in the timing chart.
  - This is the example for interface explanation, so designers can refer to make their own formats or styles.
  - For other signals in the interface list, apply the same way and strategy to make the explanation.

2.4. Design document – Implementation specification.

- Similar to interface specification, the implementation specification is created based on functional specification. The based information used to create is function partition and function interaction. From function partition, the module can be divided into sub-modules in the design and the block diagram can be extracted from function partition. Besides, the function interaction would explain how the function is implemented and the relationship between functions in the design.

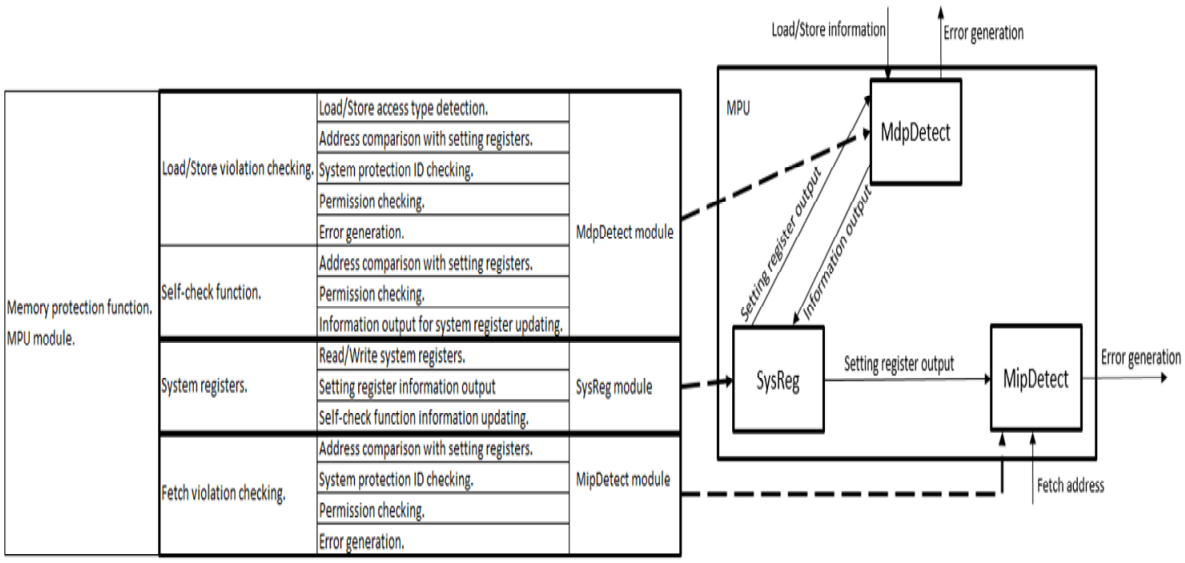


Figure 2.4.1: interface specification development from function partition.

- The contents of implementation specification are:
  - In general, the implementation specification should describe the ideas of designers for how the functions are implemented or developed rather than try to draw complex combination logic as RTL development. The latter makes readers difficult to get the author's ideas in developing the functions. Or worse, own designers find it hard to remember the reasons for the implementation when reading the document after a long time. Another drawback is the time consumption for making the detailing logic and updating design document when fixing RTL. As a result, the idea for how to implement is extremely important.
  - In implementation specification, designers usually use block diagram, true table, timing chart, or state machine to show their ideas for how to implement the functions.
  - Additionally, the implementation specification has to illustrate special implementations which are designers' intents. These special logic is essential as designers have their own reasons to apply it such as fixing bug or timing critical. Hence, it must be precisely shown for efficient verification strategy, for other designers to understand and reuse, or for avoiding similar mistakes as original designers. (refer Appendix Disappearance of design result)

- How to develop a implementation specification:
  - In general, the specification should be developed from top to down or from specific to detail. In order words, the block diagram is generated depended on function partition; then, the implementation specification illustrates the submodules as the 2.4.1 figure above.
  - The implementation specification should explain how to implement each functions in function specification. As a result, the consistency between implementation specification and function one is mandatory. It is recommended to make corresponding implementation specification based on function one.

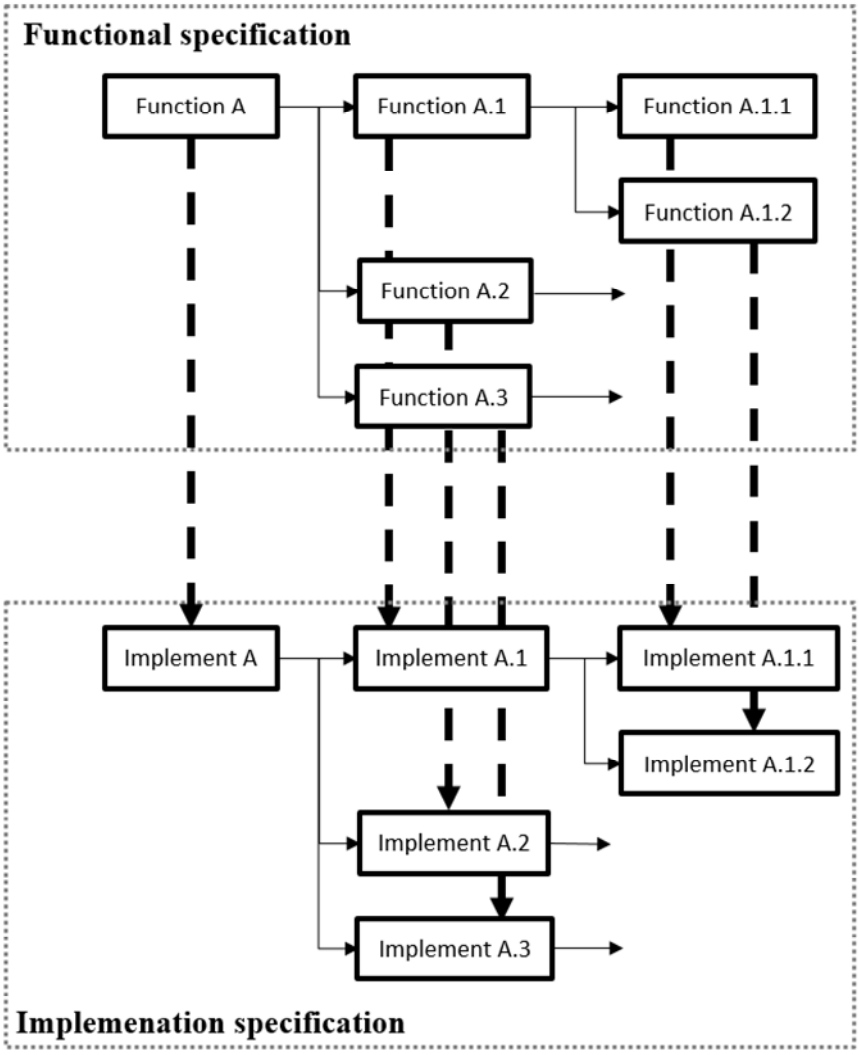


Figure 2.4.2: implementation specification develop based on function one.

2.4.1.Block diagram and Hierarchy list.

- As mentioned above, the block diagram which is the overall of a target module or design is created based on function partition. It is important to show the relationships and connections between blocks/submodule by showing the positions in the block diagram (refer figure 2.4.1).
- Besides, the block diagram helps other designers understand the structure of the design or the position of each submodule in the design.
- In some cases, the block diagram can be described following data flow if the design consists of various timing phase. For example: CPU pipeline stage diagram.

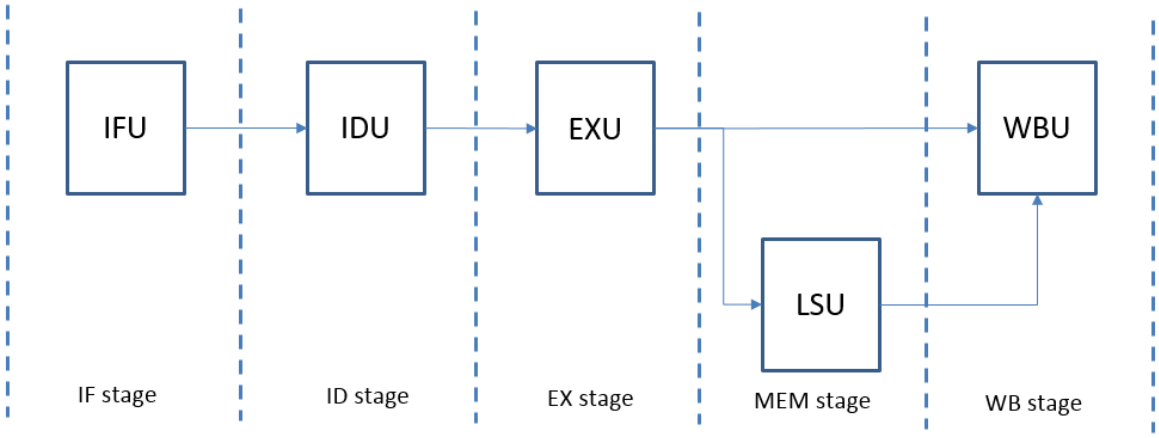


Figure 2.4.3: CPU pipeline block diagram.

- After the block diagram is made, the overall functions of each submodule should be written. The example is shown in "*Italic font*" below.

- *For a simple CPU, the implementation is divided into 5 main modules which handle the functions of each pipeline stage. Each modules is separated by FF stage to support pipeline technique.*
- *IFU main function is getting instruction code from memory and transfer the instruction code to IDU to decode.*
- *IDU handles decode function in CPU based on the instruction code from IFU. The instruction code would be the source to recognize the instruction type, register reference or immediate data. The information would be transferred to EXU to execute instruction.*
- *EXU places the role of instruction execution. In other words, EXU would execute instructions based on decoded information from IDU.*
- *LSU is used to interact with memory for data access. The instructions relating to load or store memory are operated by LSU.*
- *WBU is responsible for register file updating when the result of instruction execution is valid from EXU or LSU.*

- The hierarchy list consists of the list of submodule in the design. It is also generated based on the function partition, yet the content is module name, instance name, and the summarize functions of those submodule.

Module name			Instance name	Function
Level1(Top)	Level2	Level3		
urcp2htpua0				TPUA top level
	urcp2ltpcm0		cmn	Common control block
	urcp2ltpua0		ch0	Channel A module (channel 0)
		urcp2ntpctgf0	ctgf_a	State machine a for input capture/output compare flag control
		urcp2ntpctgf0	ctgf_b	State machine b for input capture/output compare flag control
		urcp2ntpctgf0	ctgf_c	State machine c for input capture/output compare flag control
		urcp2ntpctgf0	ctgf_d	State machine d for input capture/output compare flag control
		urcp2ntpcovf0	covf	Overflow flag generation
		urcp2ntpinc0	tpinc	16-bit incrementer
		urcp2ntpffjk0	toaq	Timer output control (JK flip-flop) a
		urcp2ntpffjk0	tobq	Timer output control (JK flip-flop) b
		urcp2ntpffjk0	tocq	Timer output control (JK flip-flop) b
		urcp2ntpffjk0	todq	Timer output control (JK flip-flop) b
		urcp2ntpincnf0	nfa	Input capture signal noise filter a
		urcp2ntpincnf0	nfb	Input capture signal noise filter b
		urcp2ntpincnf0	nfc	Input capture signal noise filter c
		urcp2ntpincnf0	nfd	Input capture signal noise filter d
	urcp2ltoub0		ch1	Channel B module (channel 1)
		urcp2ntpctgf0	ctgf_a	State machine a for input capture/output compare flag control
		urcp2ntpctgf0	ctgf_b	State machine b for input capture/output compare flag control
		urcp2ntpcovf0	covf	Overflow flag generation
		urcp2ntpcovf0	cunf	Underflow flag generation
		urcp2ntpudc0	tpudc	16-bit up/down counter
		urcp2ntpffjk0	toaq	Timer output control (JK flip-flop) a
		urcp2ntpffjk0	tobq	Timer output control (JK flip-flop) b
		urcp2ntpffjk0	nfa	Input capture signal noise filter a
		urcp2ntpffjk0	nfb	Input capture signal noise filter b

Figure 2.4.4: the example of hierarchy list.

2.4.2.True table.

- The true table shows the results of the functions based on specific inputs. Hence, the true table consists of two main part, namely input and output.
  - Input is specific conditions which are issued to the function from other logic.
  - Output is the result corresponding with each input.

Input				Output		Description
I1	I2	I3	I4	O1	O2	
0	0	0	0	0	0	Short description for each cases
0	0	0	1	1	0	Short description for each cases
0	0	1	1	1	1	Short description for each cases
0	1	1	1	1	0	Short description for each cases
1	1	1	1	0	0	Short description for each cases

Figure 2.4.5: true table example

- In some cases, additional column as “Description” should be included in true table to explain the conditions or the result of operation for each cases in true table.
- True table helps designers develop or consider the possible conditions and operations of the functions. Furthermore, verification task can be extracted from true table to avoid missing cases in verification checklist.
- In some cases, the true table is much more essential in design document rather than drawing complex logic chart. The reason is that from the specification, designers can have various coding ways and they are the personal styles of own designers. The coding styles, in some cases, are not important as the requirement is also satisfied. Thus, people does not need to cost time to make design document similar as RTL code.
- For example: from the specification, the coding styles are different, yet the result is same.

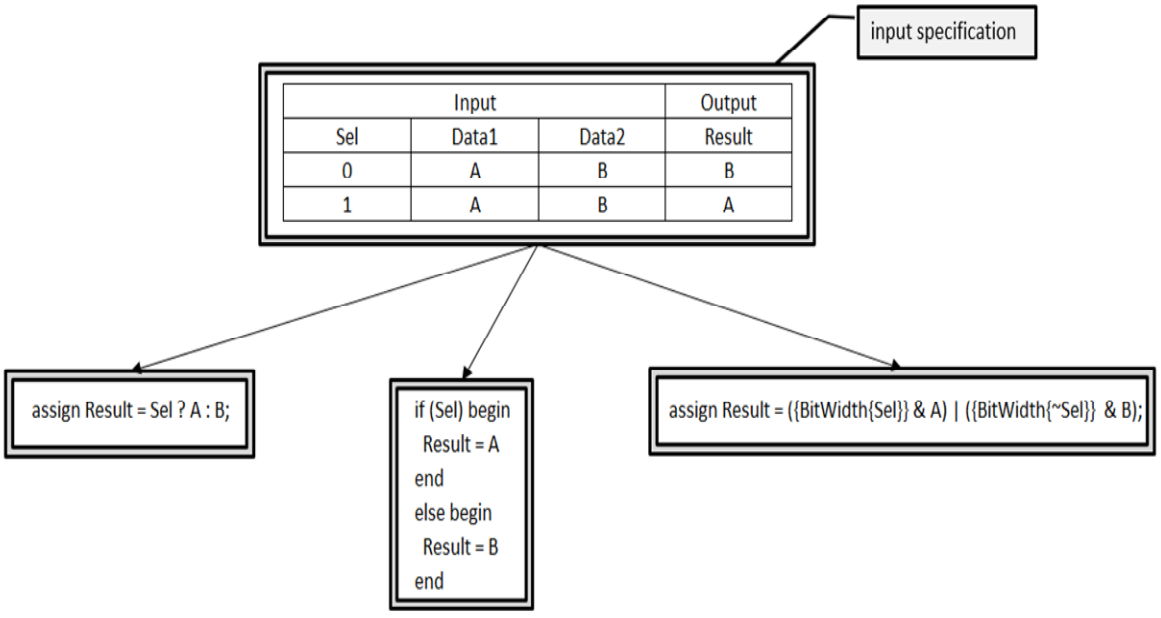
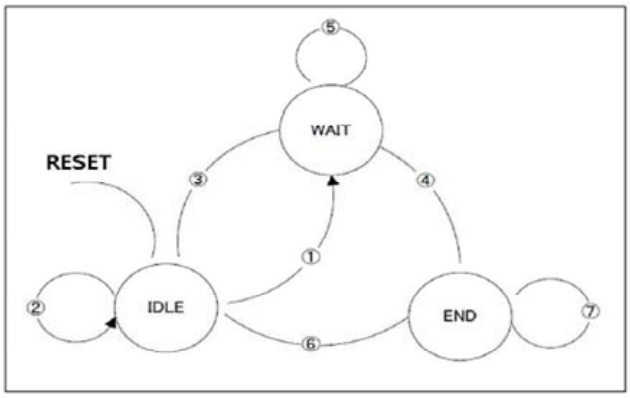


Figure 2.4.5: Example for different coding styles.

2.4.3.State machine.

- The state machine specification would show how to implement the FSM in RTL code. The FSM specification would include 2 main parts:
  - State meaning: it is a table which shows the meaning or the operation of each state in FSM.
  - State transaction: it can be a figure to show the changes in state during the operation. Besides, it is recommended to add transaction table to show the conditions to change state or explain the operation of FSM.
- Normally, the FSM implementation relates to timing phase, so the timing chart should be considered adding to show examples for the operation of FSM (Timing chart would be shown in later part).

Value	State	Meaning
00b	IDLE	Idle
01b	WAIT	Processing end wait
11b	END	Processing end



Current state	Transition condition		Transition Destination	No.
IDLE	Count start flag is ON, and count end flag is OFF.		WAIT	①
	Other than ①		IDLE	②
WAIT	Count end flag is OFF.		IDLE	③
	Other than ③	Counter value = 16	END	④
		Other than ④	WAIT	⑤
END	Counter value = 16		IDLE	⑥
	Other than ⑥		END	⑦

Figure 2.4.6: Example for state machine.

2.4.4.Timing chart.

- The timing chart is used to explain the timing in RTL design and the constraints or relationships between signals during the operation. Furthermore, as mentioned above, the timing chart also describes the state and the operation of the design in the FSM.
- To develop timing chart, the way is applied the same as timing chart in “Interface explanation”. The example below is copied from that part.

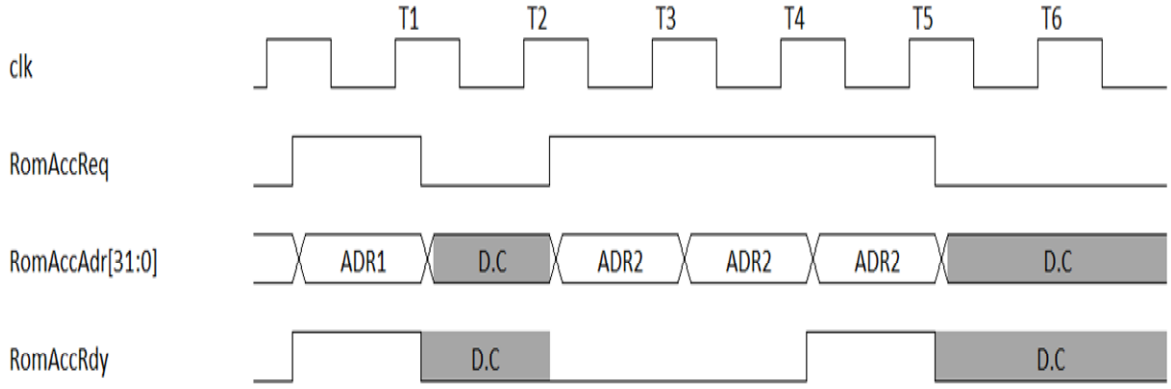


Figure <number>: the example for ICC and ROM memory protocol.

- *T1: RomAccReq issue a request to read data at RomAccAdr == ADR1, and RomAccRdy indicates that ROM memory can receive the request immediately. This is an example to show the best case of request (no pending).*
- *T2: when the request is accepted, RomAccReq signal would be de-asserted in this cycle. When RomAccReq is 1’b0, the value of address (RomAccAdr) and ready (RomAccRdy) is “don’t care”, so they can be any value. At this cycle, if the data of ADR1 are not valid, ICC cannot issue new request to ROM memory to follow the constraint of single access in ROM memory.*
- *From T3 to T4, the RomAccReq is valid, yet ROM memory is busy. Hence, RomAccRdy is 1’b0 to show the status of ROM memory. During the pending phase, the request and the address from ICC must be kept stable as the requirement of protocol.*
- *T5: RomAccRdy is asserted to indicate the availability to receive request from ICC.*
- *T6: RomAccReq is cleared, and the operation is same as T2.*

2.5. Summary.

- A good design document brings a number of benefits for the success of a project. It helps enhance the quality of the design as well as reduce the workload in design task. Furthermore, the verification can be processed with sufficient verification strategy and verification checklist thanks to the clear information in design document. In addition, a well-designed document facilitate the design legacy when designers are not original one by comprehending the information in design document. In summary, making design document costs an enormous time, yet document tasks are worth in considering the quality and success of design projects.
- The format for design document is free based on the style of designers or customer requests. This document just shows writer's style in making document to help people to get the overall requirement to make document. However, the main purpose of design document is used to explain the IDEA of designers for how to develop the functions or customer's requirements. Hence, some requirements for design document are:
  - Precise: the information in design document must be carefully written to reach the expectations from input specification and to match with the implementation.
  - Concise: the document is used to express the IDEA of designers, so it must be written for others to easily understand the IDEA of original designers. Hence, some complicated circuit logic are not recommended.
  - Clearness: the misunderstanding in reading design document may lead to serious defects for the project, so any possible misunderstanding in document should be fixed and designers must pay attention to avoiding misunderstanding descriptions in design document.
- One tip to remember when writing design document is: always try to explain the purpose of figure, timing chart, or logic circuit in the design document. Thus, others can understand what the information that writers want to emphasize is.
- Another point is how to express the designers' IDEA in English (second language). It is obvious that some designers cannot write a clear document because of the limit in English writing skills. As a result, this document introduces how to build a paragraph as a reference to help some people to improve writing skills. Although the contents may have some mistakes or limits of the author (also second language), it may help to improve English writing skills to a certain extent. If people find it necessary, please refer "Appendix Building a paragraph".

3. Appendix.

3.1. Appendix A – Report correction.

- This is the suggestion for fixing the report, so the methods may be diverse by different attitudes. Thus, all contributed opinions are highly appreciated.

Report.	Correction.
---------- **(1). Interrupt verification for RH850v2 (due day: Middle Jun)** ----------	The word "Middle Jun" is so vague, so it is better to change to specific day such as 15th Jun.
(1.1) Status: On-going – 90%	- "on-going" and "90%" cannot clearly show the status of the task. In some cases, the difficulty of remaining 10% is remarkably higher than that of 90%. Therefore, this task may be critical and need paying attention to. - In case of delay, the information should consist of delay estimate, delay reasons, delay severity, and delay action items.
(1.2) Activity: verify FEINT and DBINT interrupts.	
- Total: 419 items	
- Created: 400 items	
- Pass: 320 items	
- Fail: 80 items (7 TM)	
- Fail reason:	
o Related to PINC (5 TM): wait the available of RTL simulation.	- "Related to PINC" is quite ambiguous information because readers cannot get what is truly a problem. Ex: PINC function has not been supported or PINC function gets bugs. - There is no information of responsible side → who would be in-charge of these fail TM? - There is no information of control ticket → whether this issue is transferred to responsible side?
o STM.GSR and LDM.GSR function in Cforest has bug (2 TM).	- There is no information of control ticket → whether this issue is transferred to responsible side? - The action must be written in the report such as re-confirm with fixed-bug Cforest.
(1.3) Issue:	
- Spend time confirming old items with HVCFG.HVE=0, which costs many times due to many difficulties in fail analysis.	In this case of issue, the requirement is unclear, so what writers need must be clearly described. Ex: Spend time confirming old items with HVCFG.HVE=0, which costs many times due to many difficulties in fail analysis → need to extend 3 days or one more member to join the verification.
- Need sharing about INTC1 function.	Lack the overall situation, so readers cannot understand what the issues are. Hence, readers cannot decide which functions or which parts must be shared. Ex: Difficulty in the setting for FEINT → need to support for how to generate FEINT.
(1.4) Plan:	

- Continue create remaining TM - June 28.	
- Confirm the fail TM again with new RTL.	There is no deadline in this plan.

3.2. Appendix B – Negative condition.

- The following case is a bug detected after releasing in F1K2M product. The root cause to not recognize the issue in an early stage is the lack of verification items for negative conditions. The explanation below shows the overview of the problem in F1K2M product.
- Specification: when an instruction accessing to LRAM memory causes errors, the error information (valid signal, error address, and error type) is outputted to outside.
- From the specification, people decides to create a SVA to verify the specification. The description of SVA is
  - Condition: instruction accesses to LRAM and cause error.
  - Checking point: correctly output error information with valid signal, error address, and error type.
- Then, SVA is applied into random test, and IFV. The result for the SVA is PASS in both random test and IFV. When checking the SVA result, designers feel confident for the quality.
- However, in one special case (one special instruction), the error information output twice when one special instruction causes error. Unfortunately, the SVA above cannot detect the issue. The reason is:

	T1	T2	T3	T4
Instruction	Valid	Invalid	Invalid	Invalid
Error information	Invalid	Invalid	Valid	Valid
SVA check 1st	Matched condition		Check OK	
SVA check 2nd		Unmatched condition		Don't care

- At T1, the instruction becomes valid, so the condition to assert SVA is matched. Then, the error is outputted at T3 because the instruction cause error. In this time, SVA check and judge the result is PASS. However, at T4, the issue occurs because one more error information is outputted, but there is no valid instruction. The SVA cannot detect this case because there is no matched condition.
- Obviously, there is no bug or issue in specification writing, whereas the specification description should be updated with some negative conditions. Hence, the reviewers would possibly detect the lack of verification items.
- The description can be changed as "When an instruction accessing to LRAM memory causes errors, the error information (valid signal, error address, and error type) is outputted to outside. **Otherwise, there is no error outputted**."
- From the new description, the bug may be found when members hold the review to check the sufficiency between specification and checked items.

Specification	Checked items.	Sufficiency judgement result.
When an instruction accessing to LRAM memory causes errors, the error information (valid signal, error address, and error type) is outputted to outside.	Available - Item No: A in checklist B.	OK.
Otherwise, there is no error outputted.	None	Lack items → need to add items to verify.

3.3. Appendix C – Ambiguous description.

- The following explanation would take an example for vague specification in CPU design, which leads to an action item as improving document writing. (the detail product would be hidden for confidential reasons)
- The below information is a part of A instruction function description, which leads to the misunderstanding.
  - A instruction specification: Makes subsequent instructions wait until all the instructions ahead of this instruction have finished executing.
- In general, the description seems excellent without any problems. Nevertheless, the word “executing” leads to the fail in the conception of designers, especially CPU designers. The reason is that in CPU pipeline development, CPU operation is divided into various stages. One of the stages has the name as “executing stage”. Consequently, for some CPU designers, the word “executing” is wrongly associated with “executing stage” in CPU pipeline, which results in the wrong implementation. Taking a basic 5-stage pipeline CPU as the example.

	IF	ID	EX	MEM	WB
CPU stage			Executing stage		
Functional specification intent.					Finish all operation of an instruction.

- From the understanding of designers, A instruction affects the pipeline operation until EX stage of an instruction become finishing, while the designer for functional specification expects the influence of SYNCI would increase to the final stage of an instruction. As a consequence, a bug appears in this case.
- As a result, the description must be changed as below.
  - A instruction specification: Makes subsequent instructions wait until all the instructions ahead of this instruction have finished **their operations totally**.

3.4. Appendix D – Disappearance of design result.

- In some cases, the logic implementation is a designer's intent for special reasons, but the explanation for why the method was chosen is absent in design document. Hence, others designers may make mistakes when reusing the logic implementation without full understanding.
- The following example shows the issue of lacking description for design purposes. Fortunately, the issue is detected via synthesis phase and the harmful impact on the project is not so serious.

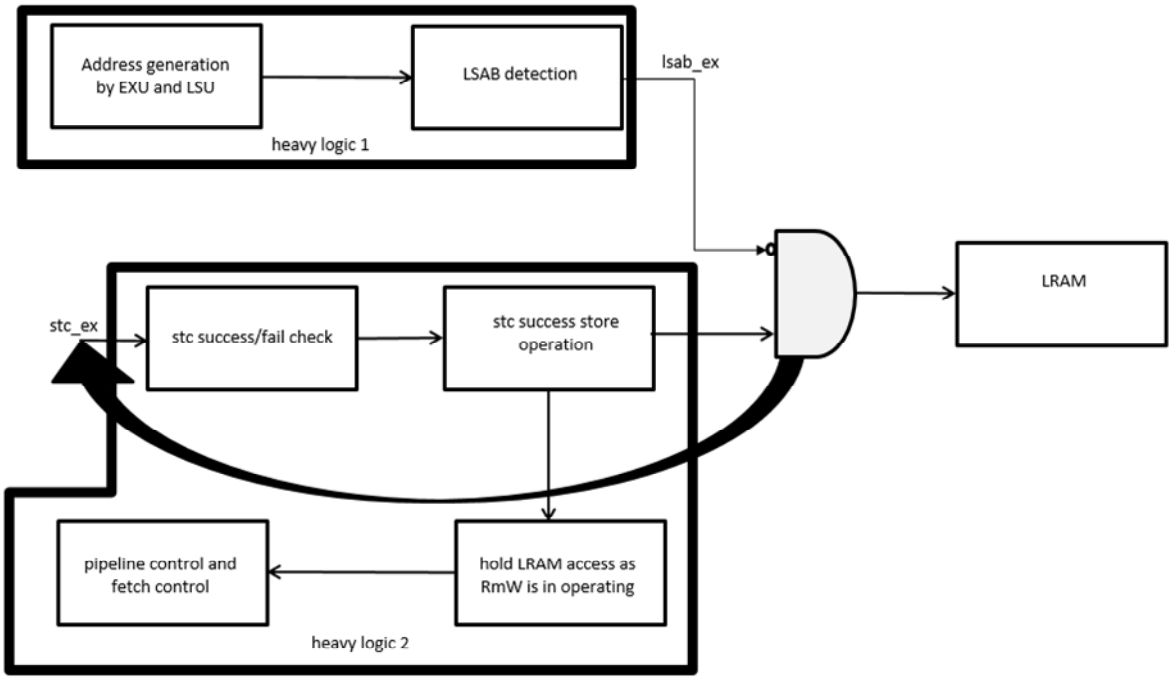


Figure D.1: Original logic implementation.

- When new engineer reuses and updates the logic implementation. The "AND" logic in original design seems to be not a good design in considering coding style, risks of bugs, and complicated control. Honestly, what new designer thinks is totally correct, so he decide to change the "AND" logic to different location (change to stc_ex as the arrow).
- On the other hand, the new designer has created a serious timing issue for the design, for the changed logic connects 2 critical timing logic. As a result, the synthesis result has a significant critical path as below.

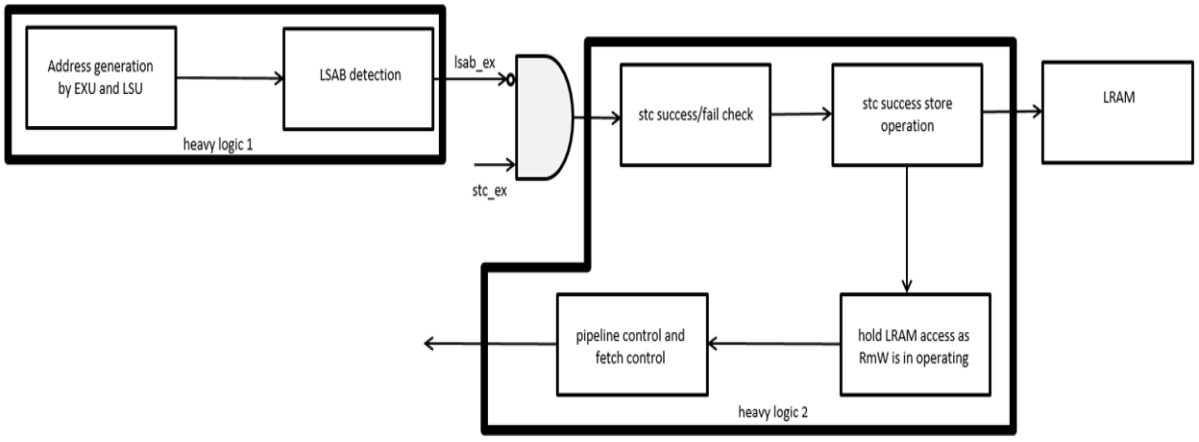


Figure D.2: Updated logic implementation.

- Even though the new implementation brings various benefits, the synthesis target cannot be achieved. The below information is the synthesis result of updated logic.

data required time	1.394400
data arrival time	-4.109767
slack (VIOLATED)	-2.715367

Figure D.3: synthesis result of critical path.

- The target timing is around 1.4, yet the violation reaches the result as -2.7. It means the real timing for this path is third times higher than the target one.
- Actually, the original designer has faced with the timing issue when implementing the function. However, the explanation for a special modification is disappeared in design document, so new designer cannot precisely understand the purpose in the location of “AND” logic.
- In conclusion, a design document which is of high quality should give information about special logics or special implementation methods for others to understand the reasons.

3.5. Appendix E – Building a paragraph.

- This part is used to explain how to write a paragraph from simple sentences. The content is basic grammar and writing in English for those who find it difficult to use second language as English in document writing.
- It is obvious that to implement a good design document, English writing skills are essential. As a result, the principle requirement for document making is English writing skill.
- Although the content may have some mistakes due to limit in author's ability, please appreciate and contribute to improving.

3.5.1. Simple sentence.

- A sentence is a group of words that usually contains a subject and a verb, and expresses a complete idea. Sentences written in English begin with a capital letter and usually end with a full stop or a question mark (https://www.ldoceonline.com/dictionary/sentence). From the description, subject (Noun or Noun Phrase) and verb are two vital elements to make a sentence, so it cannot be a sentence if either subject or verb is lacked. Hence, when writing a sentence, writer must check whether the subject and verb are available or not.
- In common sentences, it usually consists of object which is made by Noun or Noun Phrase. In some sentences, following the verb is adjective if the verb is linking verb.

S + V + O	Object is Noun or Noun phrase
S + V + Adj	Verb are linking verb. There are 11 linking verbs in English. be become remain stay appear seem sound taste feel look smell Ex: - Christmas tree LOOKS so BEAUTIFUL. - The music SOUNDS so GOOD.

- In a simple sentence, adverb and adjective are used to emphasize the meaning. Adjective normally stays before the Noun that it supports, while adverb can stay in various positions. The example below shows where adverb can be located in a sentence.

Adverb[*1], S + **Adverb**[*3] + V + **Adverb**[*4] + Adjective + O + **Adverb**[*2].

- There are 4 positions that adverb can be placed in a sentence:
  - (1): the adverb standing in this position places the role as connection between sentences. It can be called "Conjunctive adverb" (the detail would be described later.).
  - (2): one of the functions of adverb is used to support the verb in a sentence. The position is usually at the end of a sentence.
    - Ex: The power consumption in our company has increased SIGNIFICANTLY.
    - The word "SIGNIFICANTLY" is an adverb used to emphasize the verb "increase".
  - (3): Adverb standing at this position is also used to support the verb. The place of this adverb is before main verb, and after auxiliary/modal verb. What is the difference between (2) and (3) position? → Writers use adverb at (3) make clear which verb is supported in case various verbs are used in a sentence.
    - Ex: Design document must be PRECISELY matched with RTL implementation and be FULLY described all design functions.
    - PRECISELY is used to support the verb "match", while FULLY is for "describe".
  - (4): adverb is used to emphasize adjective in a sentence.
    - Ex: the description and explanation in design document are EXTREMELY important.

- Furthermore, writers also need to care about the “Tense” in the sentences to make the form of verb to show time or completion. To a certain extent, the tense that needs to use in document writing is not so much. They can be summarized as:
  - Simple past.
  - Simple present.
  - Present perfect.
  - Simple future.
  - Simple continuous.
- Above tense is common use in English writing, especially technical design document. For other types of tense, people can investigate and improve later if they have time.

3.5.2. Complex/Compound sentence.

- Complex/Compound sentence is made up by more than one simple sentence. As a result, they would have at least 2 subjects and 2 verbs in a sentence.

* Note: people can google complex and compound sentences for the differences between them. From now, the word "complex sentence" is used to indicate a sentence that have more than one clause.

- Because complex sentences consists of various simple sentences, they need conjunctive words to connect these simple sentences. The common uses of conjunctive words are coordinating conjunctions and subordinating conjunctions.
- Coordinating conjunctions can be easily remembered by the word "FANBOYS".

Coordination	Description.
For	It is used to express the rationale. It has the same meaning as "because". Ex: I do not eat meats, FOR I am a vegetarian.
And	It is used to show the non-contrasting ideas, items, or opinion. Ex: Making document brings benefits for design phase, AND the quality of verification phase is improved.
Nor	Nor expresses non-contrasting negative ideas. Ex: They do not gamble, nor do they smoke.
But	It presents for the contrast or exception. Ex: One redundant error information is outputted to outside, BUT there is no valid instruction in CPU.
Or	Or illustrates the alternative items or idea. Ex: Designers can use FSM to explain transaction state, or they can use timing chart instead.
Yet	Yet has the meaning as But. However, in some cases, Yet is used to express an unexpected result. Ex: Time consumption for document making is enormous, yet it helps to reduce the time in design phase.
So	So shows the result or consequence. Ex: The design document has not described the combination list, so some verification items are lacked.

- As described above, writers can use subordinating conjunctions to make complex sentences. However, the number of subordinating conjunctions is various. Hence, in this document, some common conjunctions are mentioned for reference only.

Subordination	Description
Although/Even though/Though/Whereas	Show the contrast.
After/Before/Since/Until/When/Whenever/While	Show time.
If/Unless/As long as/Providing that/Provided that	Show conditions.
Because/As	Show the reason.

- Ex:
  - Because Mary and Samantha arrived at the bus station before noon, I did not see them at the station.
  - While he waited at the train station, Joe realized that the train was late.
  - After they left on the bus, Mary and Samantha realized that Joe was waiting at the train station.

* Note: Relative clause is one of the way to make complex sentence. People can easily find structures or how to use relative clause, so it is omitted here.

3.5.3.Building a paragraph.

- A paragraph is a group of several related sentences in a piece of writing, with the first sentence beginning on a new line. Because a paragraph is a combination of various sentences, the main requirements for a paragraph are “coherence” and “cohesion”.

o Cohesion: it requires that the idea in a paragraph must be unit. In other words, one paragraph would illustrate one main idea, so the sentences in that paragraph would mention or support that main idea (Normally, the main idea locates on the first sentence of a paragraph, namely topic sentence).

	Bad	**Good**
Example.	*Making **design document** brings various **benefits** for the **following phases**. The first requirement for design document is equivalent in quantity between design document and RTL implementation to ensure the consistency. Secondly, the quality of design document should be paid attention to avoiding potential bugs.*	*Making **design document** brings various **benefits** for the **following phases**. Firstly, design document helps designers consider the structure of RTL implementation at the early phase. Secondly, from design document, verification engineers can define efficient verification strategy and sufficient verification items.*
Analysis.	The topic is benefits of design document for later phases. However, the supporting sentences indicate the requirements of design document (quality and quantity). As a result, this paragraph lacks the cohesion.	The topic is benefits of design document for later phases. Hence, the following sentences should illustrate what are the advantages of design document. In this example, the benefits for design and verification are mentioned.
Example for cohesion in a paragraph.		

o Coherence: as mentioned above, a paragraph may consist of a number of sentences, so these sentences should be connected together to show or support the main idea in the paragraph. As a result, “coherence” can be defined as how to link the sentences in a paragraph. The table below introduces some conjunctive adverb to connect or transit the sub ideas in a paragraph.

To show additional information.	Also, in order words, for example/instance, moreover, furthermore, in addition, additionally, to be more precise, to be more details, more importantly, besides.
To show the contrast or opposite idea.	However, nevertheless, nonetheless, although, on the other hand, in contrast, by contrast, but, in spite of, instead.
To show the result or conclusion.	Hence, therefore, thus, in conclusion, finally, as a result, as a consequence, consequently, summarily.
To refer an idea/things already presented.	Use pronoun as this, it, that, those, these.
To introduce a number of point.	The first point is, there are various ways/means/methods, another point is, second/third point is.
To show the sequence of idea.	Firstly/Secondly/finally, first of all, to start with, then, after that, next, as soon as, immediately, suddenly, unexpectedly, in the end.
To show the face or true.	Obviously, undoubtedly, definitely.

Sentences in paragraph	Explanation.
Making design document brings various benefits for the following phases.	This is the topic sentence which mentions the benefits of making design document.
Firstly, design document helps designers consider the structure of RTL implementation at the early phase.	"Firstly" is used to show the first benefit of design document.
Hence, it causes less modifications in RTL design phase for implementation structure because of well-organized design document.	"Hence" is used to support the idea of RTL design benefits from making design document.
Secondly, from design document, verification engineers can define efficient verification strategy and sufficient verification items.	"Secondly" is used to show another idea or benefit from design document making.
In other words, from design document, people can understand the implementation, so they can define how to verify the functions and which items should be included or removed due to unreachable cases.	"In other words" signals for the additional explanation to support the idea of effective verification activities.
As a result, making design document is worth as its benefits, although the workload is huge.	"As a result" is used to show the conclusion for the idea of design document's benefits.

- From the example above, these sentences in a paragraph are connected to keep the cohesion. If there is no conjunctive adverb, these sentences are disjoined, which causes difficulties for readers to link the ideas in the paragraph together.
- For a well-organized paragraph, it is recommended to follow the flow below.

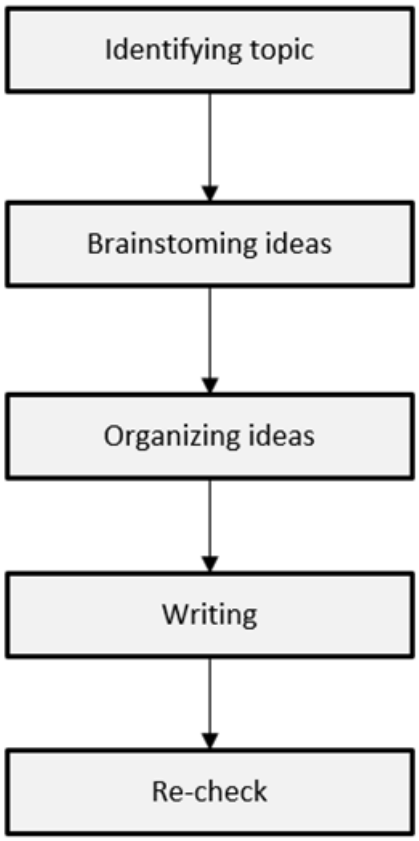


Figure E.1: Paragraph writing flow.

- Identifying topic: once authors write a paragraph, they must identify what is the main purpose or main meaning of that paragraph. It is the principle requirement as all sentences in a paragraph must be written to focus on that topic, which would follow the cohesion requirement for a paragraph.
- Brainstorming ideas: writers try to think and consider how to develop the paragraph from main idea. In other words, when main idea is given, people must try to show the evidence, explanation, or example to support or prove the main idea. Normally, for design document, they are usually explanation for the reason of implementation.
- Organizing ideas: this step is used for writers to consider how to structure the paragraph. It means how the main idea and sub ideas are arranged in a paragraph. Generally, the main idea should be written at the first sentence which is called topic sentence. The topic sentence would show the overall meaning of the paragraph and help readers recognize the main meaning of that paragraph. Besides, how to put sub ideas or explanations in a paragraph should be carefully considered to make sentences or ideas logically link together. This step is the most important part in paragraph writing.
- Writing: writers start to write a paragraph based on the results of above steps. In this step, full sentences should be written, and how to connect sentences in a paragraph is applied.
- Re-check: this is the final step when full paragraph is made. Writers should use a little time to re-check their writing to correct grammatical mistakes, sentence structures, word choice, and so on. It helps to improve the quality of paragraph.

* Note: (these comments are author's opinions and they are just for references.)

- The content in this part is not all skills that need to write a paragraph. It is just simple ways to help those who are not good at English writing form a general mind for how to write a paragraph. In general, people should try to write simple sentences, and complex sentences. Then, people can try to link or connect these sentences together to make a paragraph.
- As mentioned above, to write design document, the number of tense which can be used is limited, so people can focus on learning and effectively using these tenses first rather than trying to study various tenses in English.
- To improve writing skills, it requires a hard work and people should write as much as possible. However, it is better to have someone to help to correct writers' mistakes or to improve the writing as the mistakes would be repeated if they are not identified.
- TOEFL and EILTS are immediate English certificate that consists of writing skills. It is recommended that people should take part in these English course to improve their writing skill, although the writing mainly focus on academic writing.
- One of the most important points for writing is vocabulary. It is recommended to study vocabulary with its synonym, antonym, and word family. One of online dictionary that has a rich content for vocabulary is Longman. It can be accessed by https://www.ldoceonline.com.
- Another reference document for writing is "How to write an effective business document – DungVuu <dung.vuu.xa@rvc.renesas.com>".

4. History.

Date	Author	Part	Description
2017/12/07	VuHuynh	-	New creation.
2018/02/01	VuHuynh	2.1	Figure 2.1.1: change order of interface and implementation specification.
		2.2.3	Figure 2.2.3/2.2.4: change the example as real implementation.
		2.3	Figure 2.3.1: change the example as real implementation.
		2.4	Figure 2.4.1: change the example as real implementation.
2018/05/10	VuHuynh	1.2	Add severity estimation into the delay status.
		1.5	Add risk prediction into “Plan” and add severity for risk prediction.