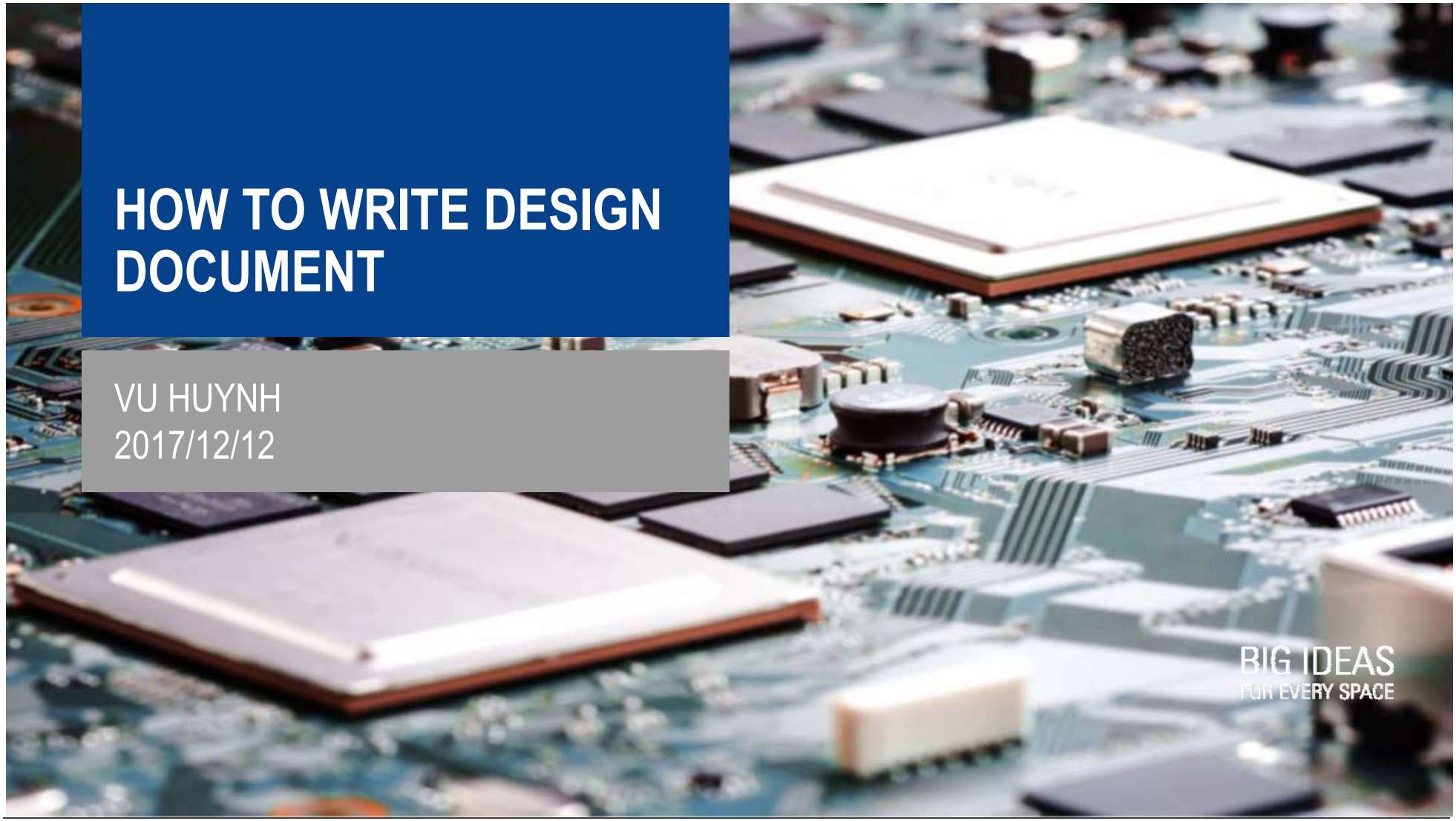
HOW TO WRITE DESIGN DOCUMENT

VU HUYNH
2017/12/12

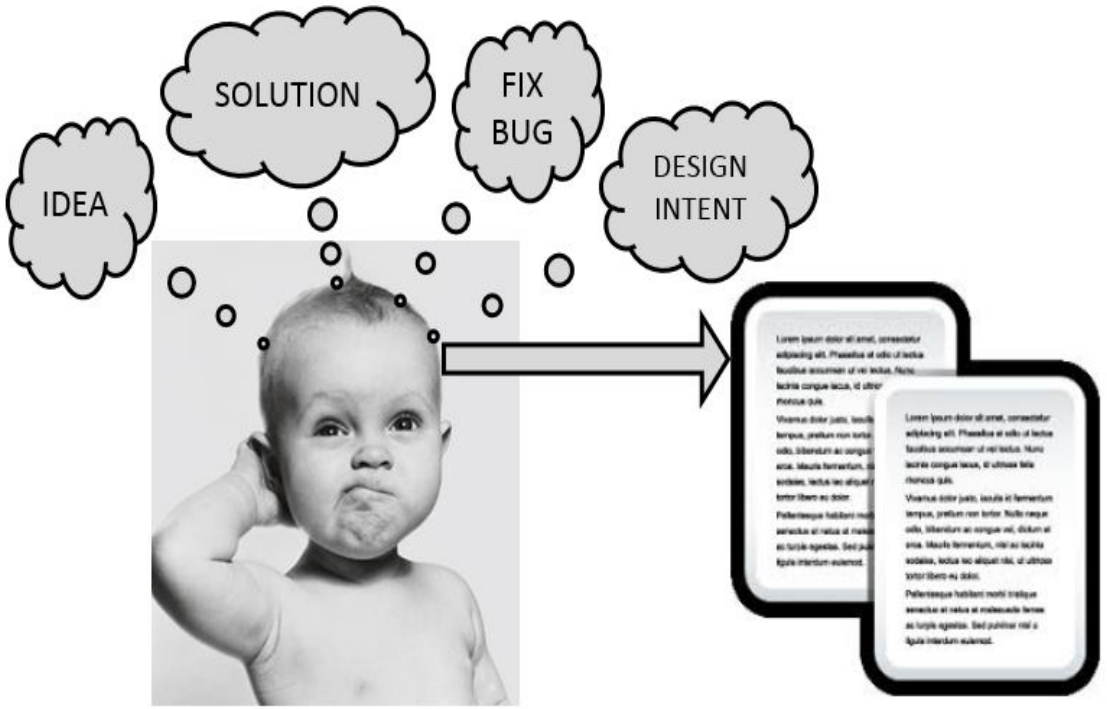


0. OVERALL.

1. Design document – Introduction.
2. Design document – Partition.
3. Design document – Functional specification.
   - 3.1 Function description.
   - 3.2 Function list.
   - 3.3 Function interaction.
   - 3.4 Function partition.
4. Design document – Interface specification.
   - 4.1 Interface list.
   - 4.2 Interface explanation.
5. Design document – Implementation specification.
   - 5.1 Block diagram and hierarchy.
   - 5.2 True table.
   - 5.3 State machine
   - 5.4 Timing chart.
6. Summary.

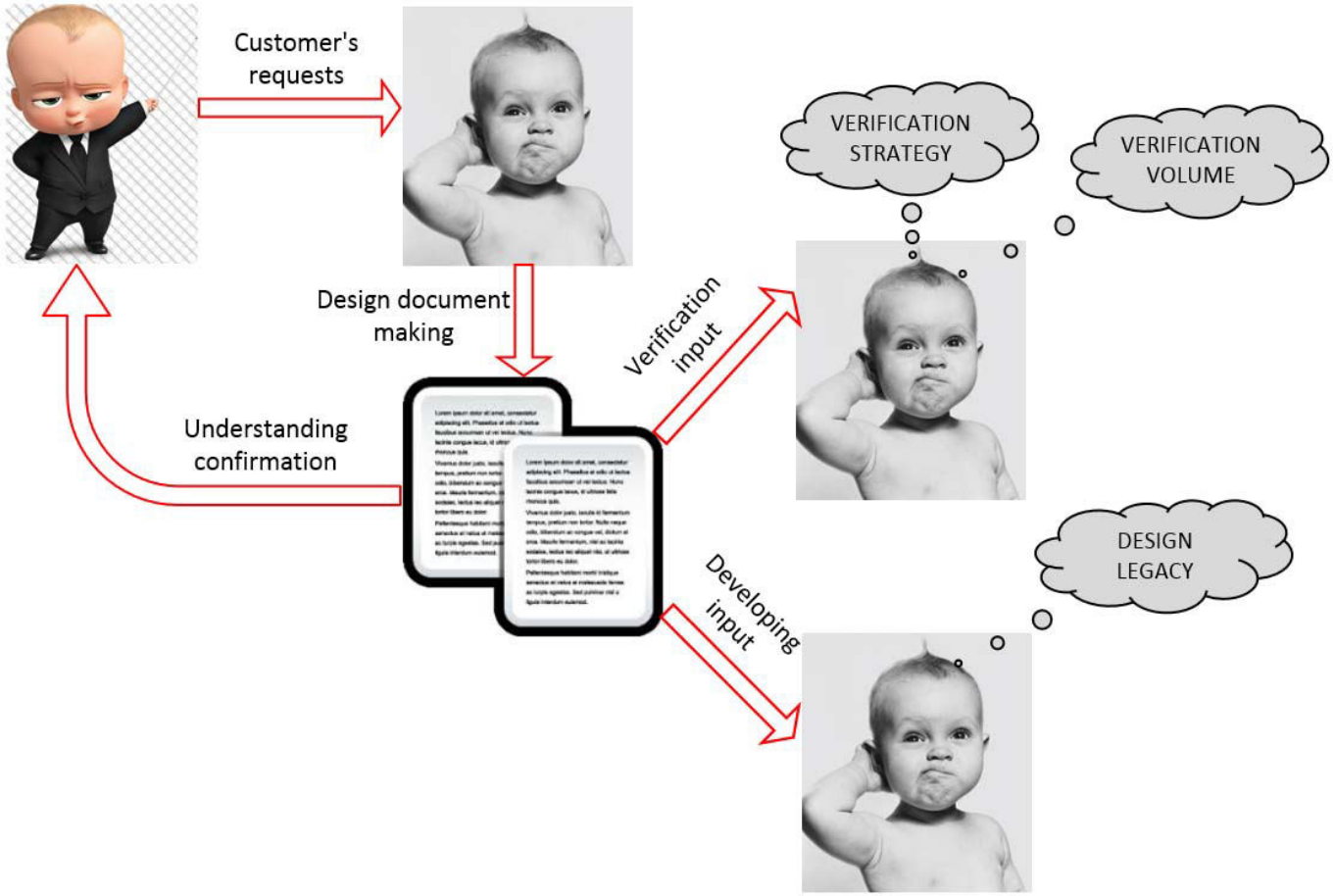
1. DESIGN DOCUMENT – INTRODUCTION.

- What is design document?
  - ➢ Design document is created to explain how a microprocessor or IP are made.
  - ➢ The content in the document may be:
    - ✓ How a microprocessor or IP are made.
    - ✓ Why the methods are choice.
    - ✓ What are the design intents.
    - ✓ What are the problems if not follow the ways.
    - ✓ What are the benefits by applying the ways.



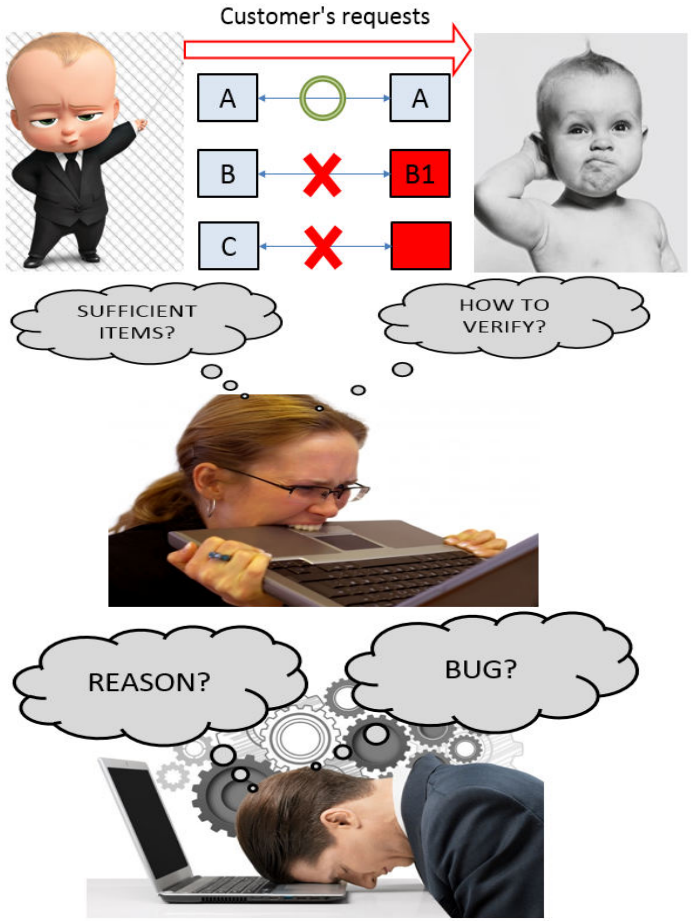
1. DESIGN DOCUMENT – INTRODUCTION.

- What are the main functions of design document?
  - Confirm the understanding of designers for input specification with customers.
  - Facilitate verification efficiency which reduces lacking items and improve verification strategy.
  - Improve understanding of reusing designers to avoid possible bugs.



1. DESIGN DOCUMENT – INTRODUCTION.

- What are the drawback if no design document?
  - ➢ Misunderstandings or missing expectations from designers for customer's requests.
  - ➢ Difficult for verification phase as how to verify or how much verification items are enough.
  - ➢ Reusing designers cannot understand the reasons of implementation or how to implement the functions, which may cause bugs in modification.

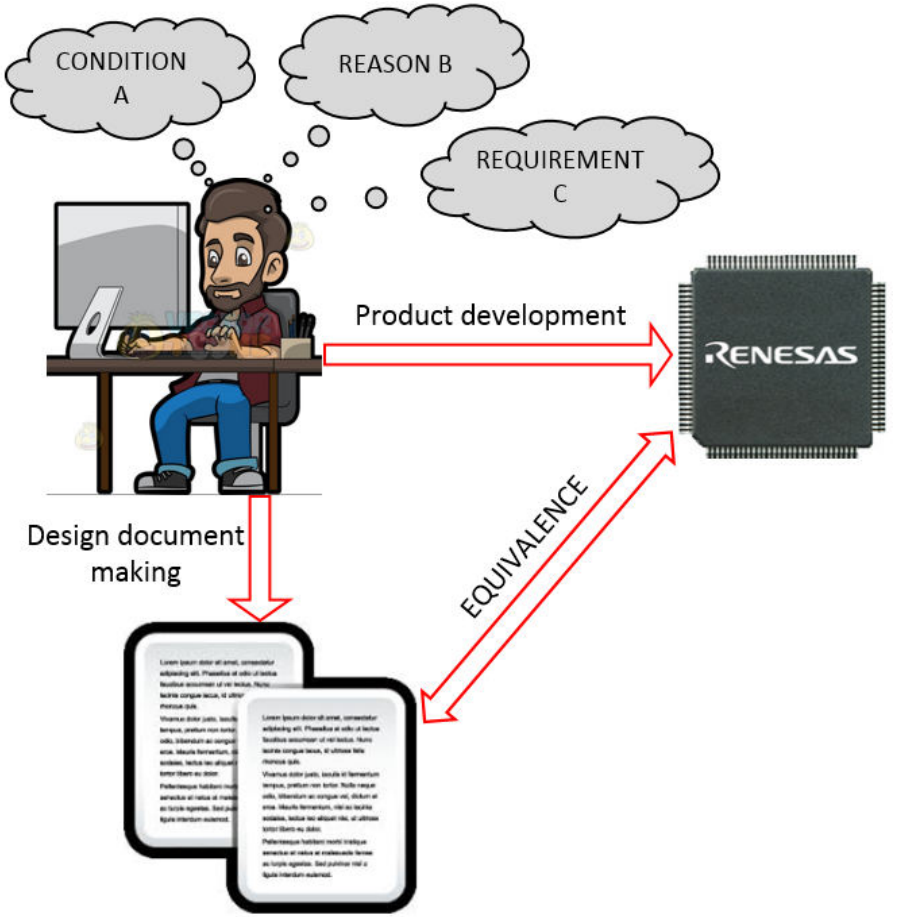


1. DESIGN DOCUMENT – INTRODUCTION.

- What are the requirements of design document? – Quantity and Quality
  - Quantity:
    - ✓ Adequacy.
    - ✓ One document.
    - ✓ Conciseness.
  - Quality:
    - ✓ Consistency.
    - ✓ Design intents.
    - ✓ Clear description.
    - ✓ Negative conditions.

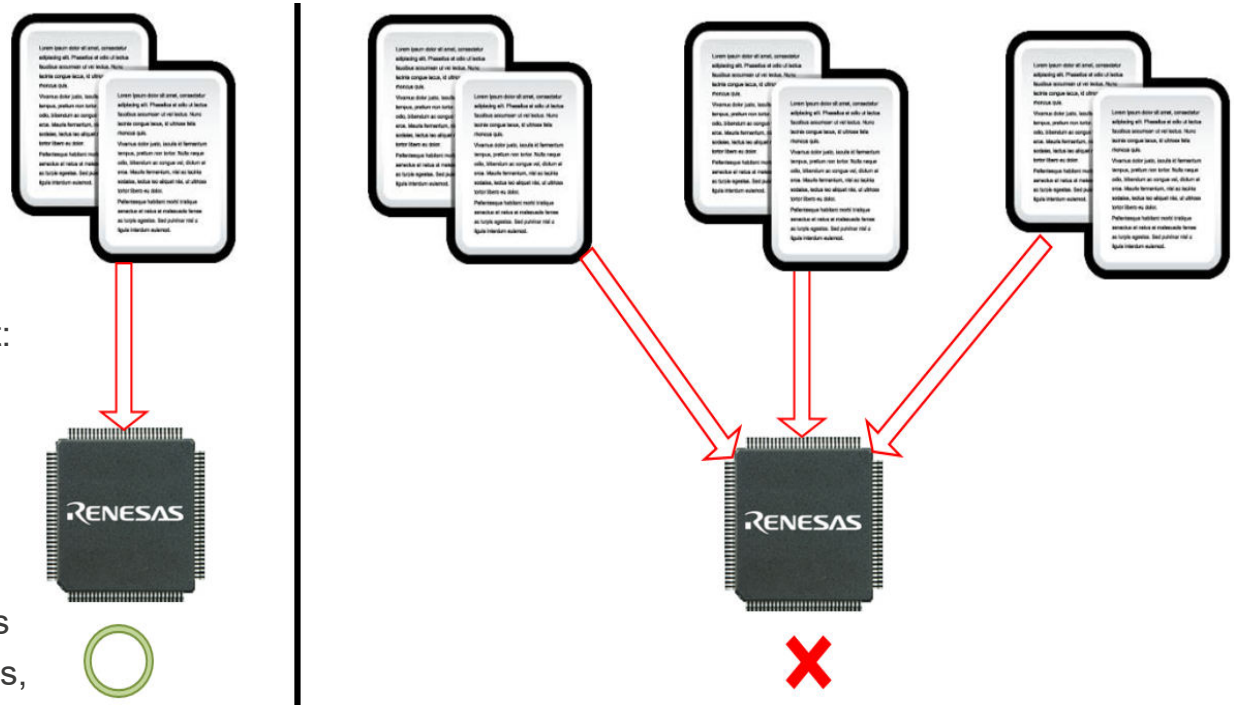
1. DESIGN DOCUMENT – INTRODUCTION.

- Quantity:
  - ➢ Adequacy:
    - ✓ Conditions for the implementation must be enough.
    - ✓ Descriptions of the implementation must be enough.
    - ✓ Features of the implementation must be enough.
  - ➢ In conclusion, what are needed for the implementation must be described in design document.
  - ➢ Negative impacts of inadequacy:
    - ✓ Insufficient implementations because of lacking conditions.
    - ✓ Insufficient verifications because of lacking items.



1. DESIGN DOCUMENT – INTRODUCTION.

- Quantity:
  - ➢ One document.
    - ✓ All things that need for the implementation should be put into one document.
  - ➢ Negative impacts with various document:
    - ✓ The possibility to forget to refer information in other documents → unmatched requirements, implementation bugs, and so on.
    - ✓ The possibility to lack verification items in other documents → undetected bugs, insufficient verification items, etc.

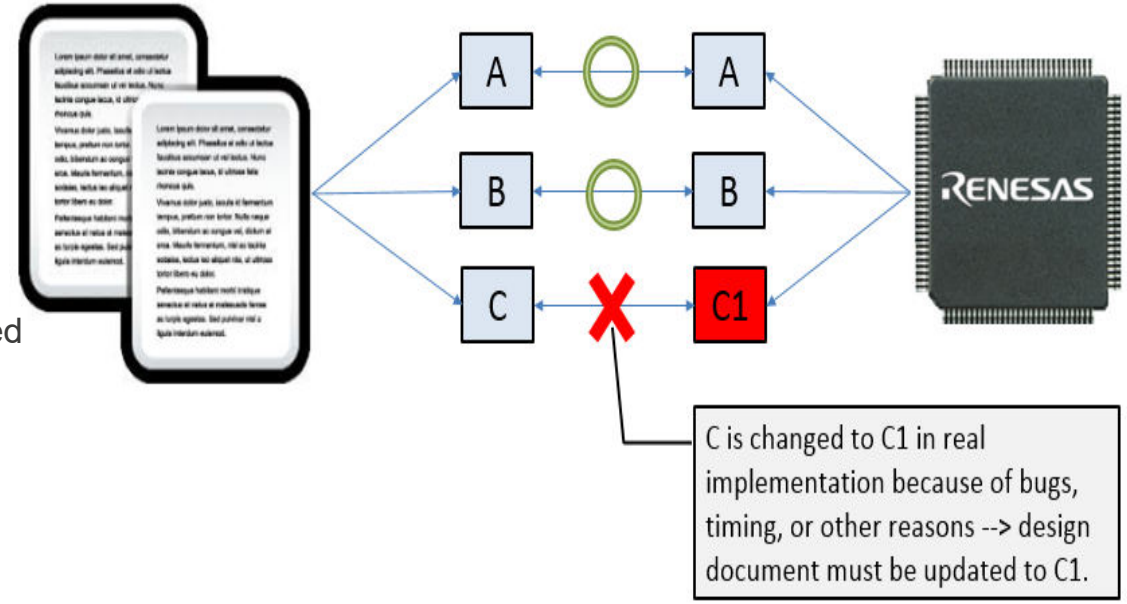


1. DESIGN DOCUMENT – INTRODUCTION.

- Quantity:
  - ➢ Conciseness.
    - ✓ Only necessary information should be described.
    - ✓ Redundant information should be removed from document.
  - ➢ Negative impacts with messy information:
    - ✓ Redundant information may cause confusing for readers.
    - ✓ Redundant information may hide the main ideas, or information.
    - ✓ Redundant information leads to difficulties when searching main information.

1. DESIGN DOCUMENT – INTRODUCTION.

- Quality:
  - ➢ Consistency.
    - ✓ The information in design document must be correct and equivalent to RTL design.
    - ✓ During the development phase, RTL design may be changed because of bugs or other reasons → design document must be updated as well.
  - ➢ Negative impacts of inconsistency.
    - ✓ Readers can hardly understand the implementation via specification.
    - ✓ Verification items may be incorrect.
    - ✓ Potential bugs may occur in case of design modifications.

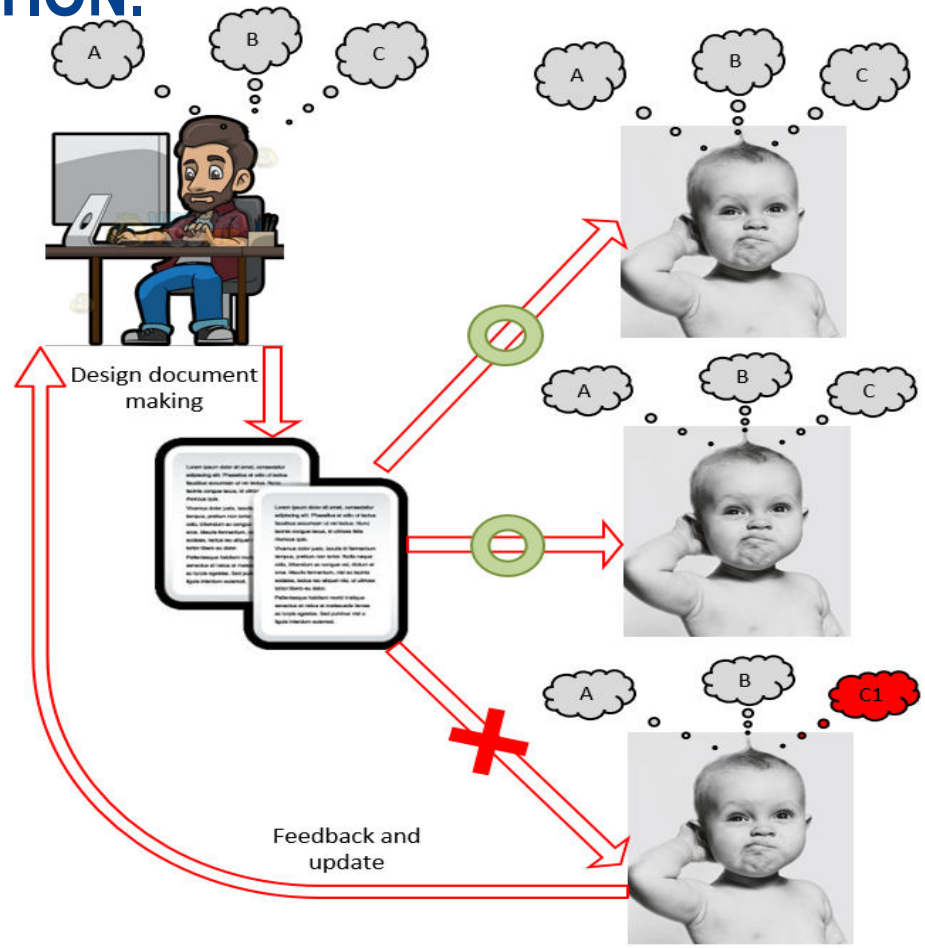


1. DESIGN DOCUMENT – INTRODUCTION.

- Quality:
  - ➢ Design intents.
    - ✓ Explain the purposes or reasons why the implementation is chosen.
    - ✓ Explain the contents of the design such as the meaning of the logic, the special points of the table, or the focusing points of the timing chart.
  - ➢ Negative impacts of missing design intents:
    - ✓ Other people may not recognize the main points.
    - ✓ Mistakes/bugs may occur due to misunderstand the design.
    - ✓ Same mistakes may happen.
  - ➢ Note: real example is in “HowToWriteDocument.docx – Appendix: Disappearance of design result”.

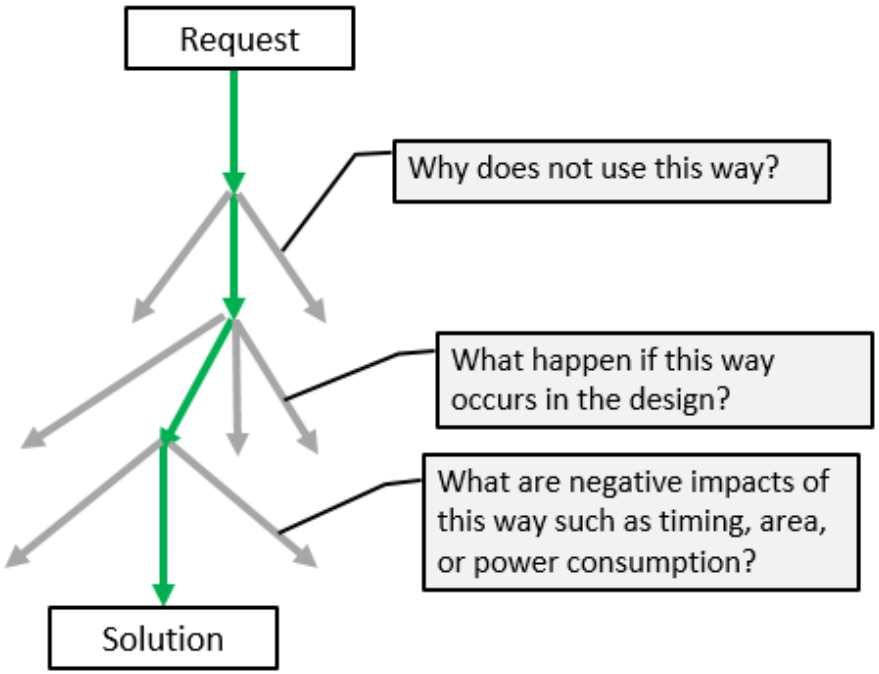
1. DESIGN DOCUMENT – INTRODUCTION.

- Quality:
  - ➢ Clear description.
    - ✓ The description must be clear for everyone to read and understand as the same way.
    - ✓ Design document is used for various purposes (review, development, or verification) → mistakes cause serious problems.
    - ✓ Document must be updated whenever anyone understands in a different way.
  - ➢ Negative impact of unclear description.
    - ✓ Misunderstanding in verification → hidden bugs.
    - ✓ Misunderstanding in reusing implementation → new bugs.
  - ➢ Note: real example is in “HowToWriteDocument.docx – Appendix: Ambiguous description”.



1. DESIGN DOCUMENT – INTRODUCTION.

- Quality:
  - ➢ Negative conditions.
    - ✓ Explain what happen if not follow.
    - ✓ Explain what happen if conditions are false.
  - ➢ The workload for this job is enormous, but it should be done to reduce the risk of undetected bugs.
  - ➢ Negative impacts of unexplained negative conditions.
    - ✓ Readers may not understand what will happen if not follow.
    - ✓ Reusing designers may make bugs in case of negative conditions.
    - ✓ Verification members may miss the negative items.
  - ➢ Note: real example is in “HowToWriteDocument.docx – Appendix: Negative condition”.

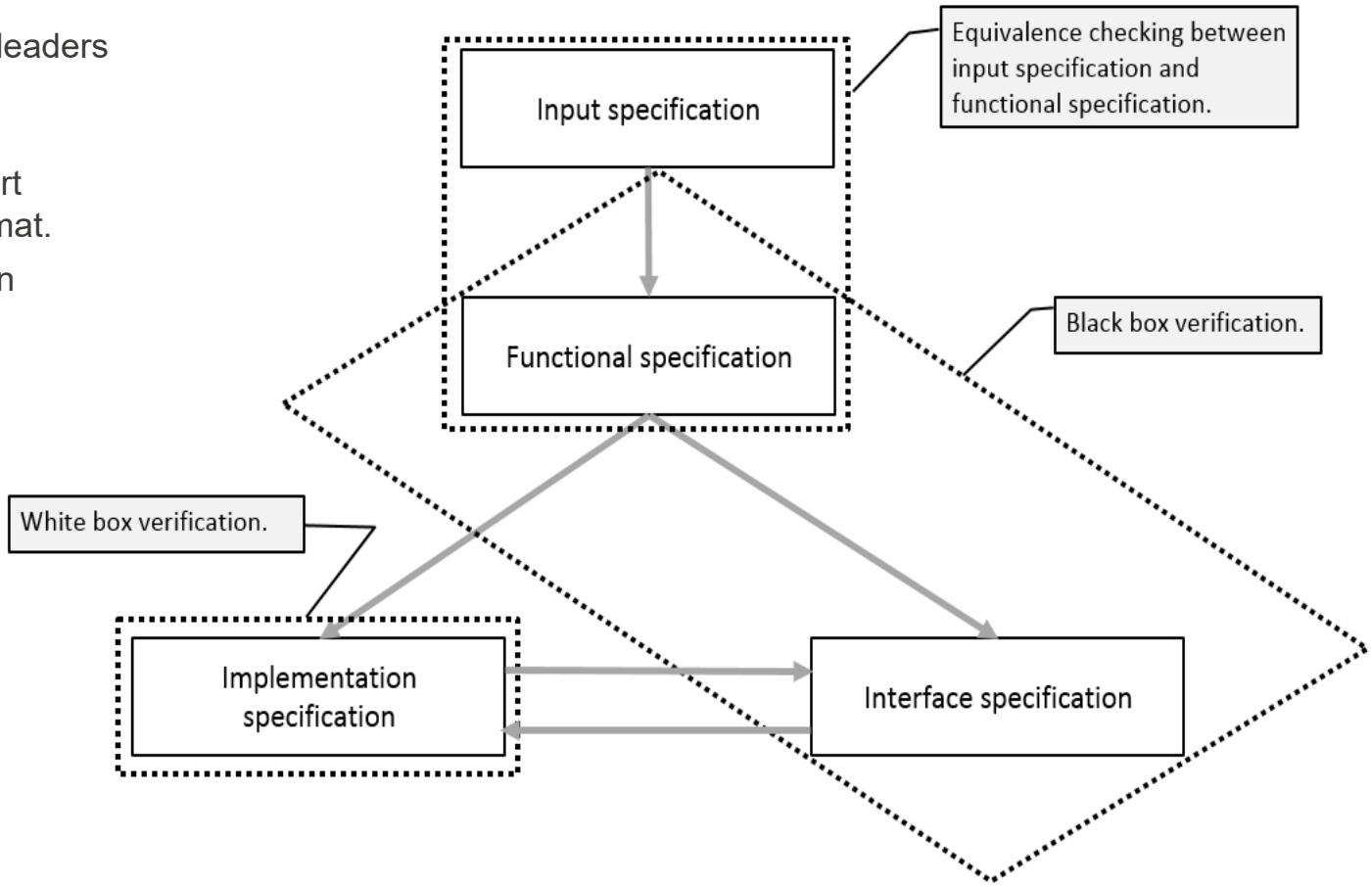


2. DESIGN DOCUMENT – PARTITION.

- The format of design document must be defined.
- The advantages of format:
  - Principle requirements for a design document (make by experience designers - useful for new designers).
  - Well-organized information in a design document.
  - Avoiding messy writing styles from various designers.
- How to define a format for a document?
  - Top level to bottom level.
  - General functions to detailing functions.

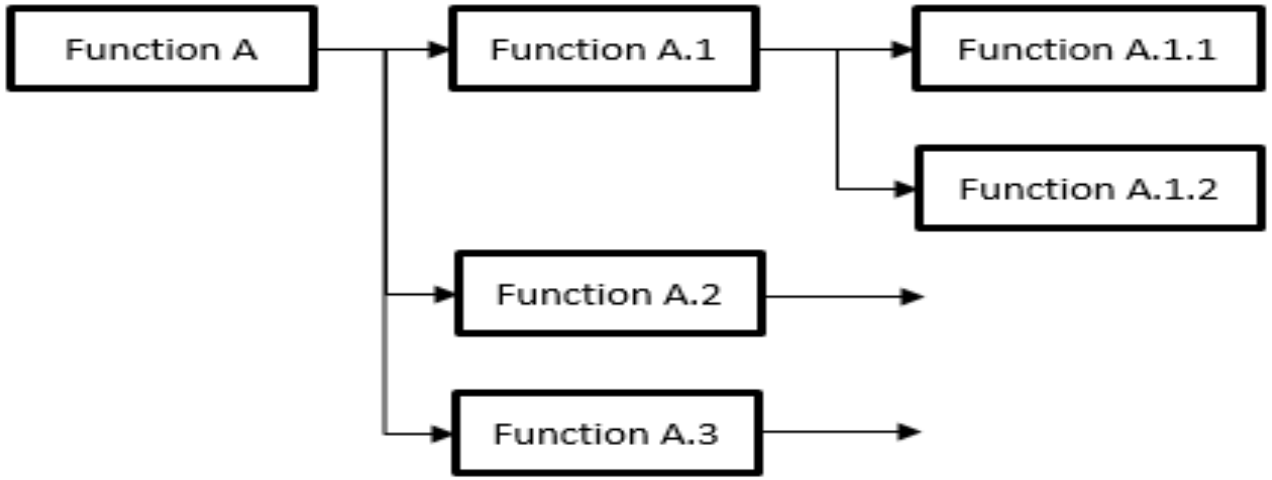
2. DESIGN DOCUMENT – PARTITION.

- The format depends on design leaders or customer requests.
- The following format is writer's suggestion and the following part would explain based on the format.
- There are 3 main parts in design document.
  - ➢ Functional specification.
  - ➢ Interface specification.
  - ➢ Implementation specification.



2. DESIGN DOCUMENT – PARTITION.

- Input specification is used to create functional document.
- Functional document lists up all functions in the design needed to satisfy customer requests.
- Functional document usage:
  - The foundation to develop interface specification and implementation specification.
  - Black box verification application.
  - Sufficiency checking between input requests and supported functions.

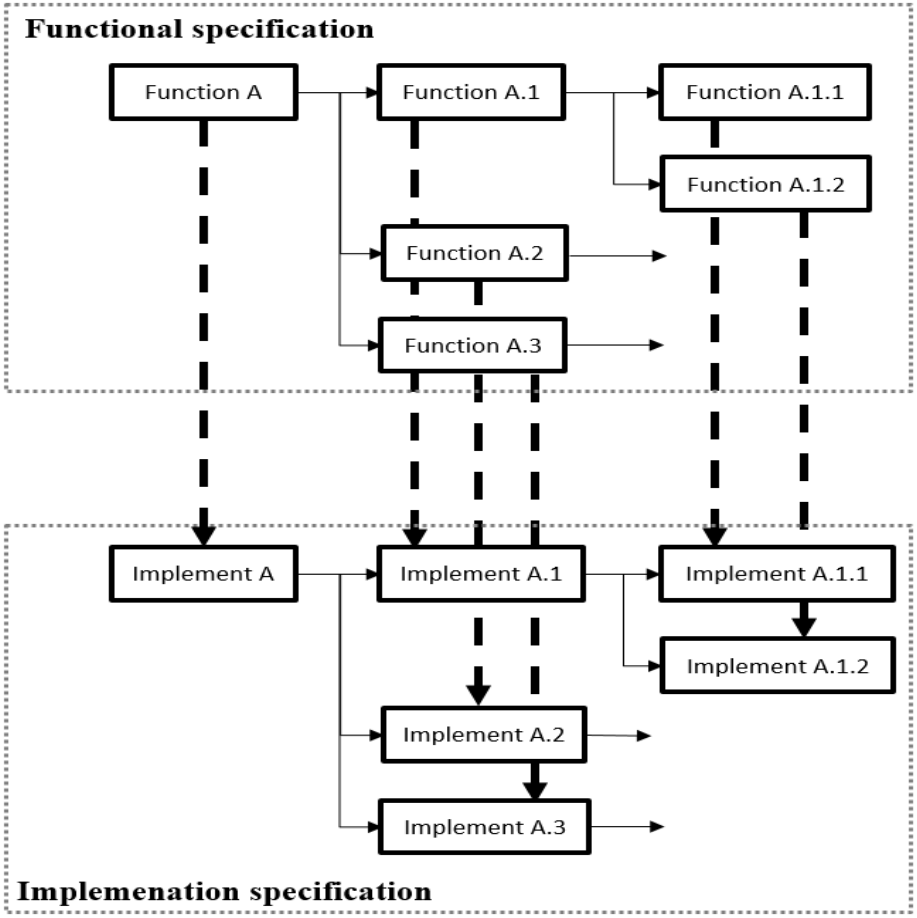


2. DESIGN DOCUMENT – PARTITION.

- Interface specification is made from functional specification.
- Interface specification mainly describes the protocol between modules, timing sequences, or expectations of input and output.
- Consider following points to make interface specifications for a function:
  - What are needed to implement the function.
  - When the input should be valid.
  - What information should be outputted.
  - When it should be outputted.
- Interface specification usage:
  - Apply in black box verification associated with functional specification.
  - Provide protocol between connected units or modules.

2. DESIGN DOCUMENT – PARTITION.

- Implementation specification is made from functional specification and interface specification.
- The main purpose of implementation specification is how to implement a function in detail.
- Each function in functional specification must be mapped with a part in implementation specification.
- Implementation specification usage:
  - ➢ Confirm the implementation methods in the early phase.
  - ➢ Use in white box verification.
  - ➢ Facilitate the legacy for later development.



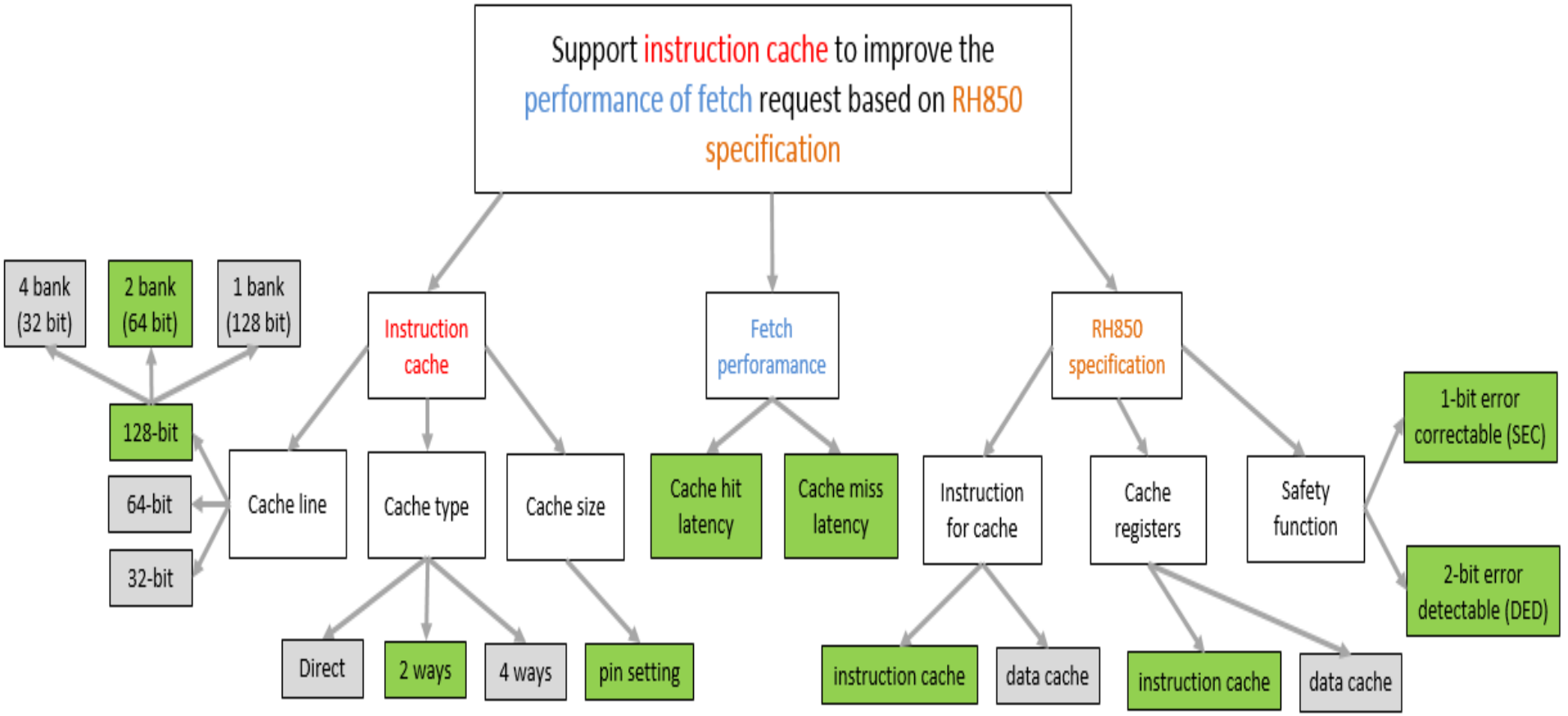
3. DESIGN DOCUMENT – FUNCTIONAL SPECIFICATION.

- As mentioned above, functional specification is made from:
  - Customers' requests.
  - Standard specification such as IEEE or AMBA.
- Functional specification should show how to reach input specification.
- Functional specification's benefits:
  - Confirming the understanding of designers for input requests.
  - Facilitating the reuses for other designers.
  - Organizing the design.
  - Reducing missing verification items.
  - Generating complex items which are combined from various functions.
- Functional specification contents:
  - Functions description: the overall ideas to adapt customer requests.
  - Function list: list of necessary functions from top levels to bottom levels.
  - Function interaction: the relationships between functions in the design.
  - Function partition: grouping related functions together.

3.1. FUNCTION DESCRIPTION.

- Function description is created by listing up necessary functions for reaching the input specification.
- The reason for function description:
  - Equivalence checking between design specification and input specification.
  - Consideration for how to satisfy the requirement.
  - Foundation for making later part.
  - Overall concepts of the design.
- Note: the function description introduces the overall functions in the design, so designers should not put so detailing implementation into this part.

3.1. FUNCTION DESCRIPTION.



3.1. FUNCTION DESCRIPTION.

- From the mind map, the function description can be written 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: Support parity error detection for address, single bit error correctable for data, and double bit error detection for data.

3.2. FUNCTION LIST.

- Function list is a table or list that shows detailing functions in the design.
- The list shows top functions to bottom functions.
- The input to make function list is items in function description.
- Ex:
  - One item in function description can be considered as first class.
  - Second class functions are created to explain how to support the function first class.
  - Third class functions are created to explain how to support the functions in second class.
  - This way would be applied until n class.
- Note:
  - The number of n class may depend on the complex of the design.
  - The number of n class may depend on the format of design document or request from customers.
  - Normally, the number of n class may be three or four to illustrate a design.

3.2. FUNCTION LIST.

- The format of function list may be made as the table below.

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

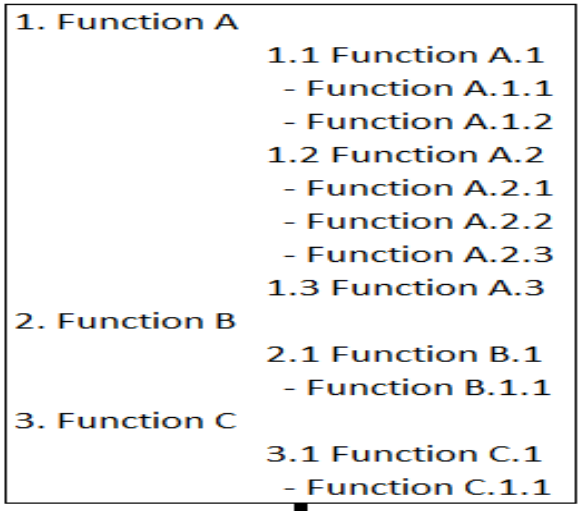
3.2. FUNCTION LIST.

- Ex.

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.

3.2. FUNCTION LIST.

- In case of complicated design, the function list can be written as tree functions as normal document.
- Each first class is put into one section. Then, the section is developed into second class and third class.
- However, for later uses (function interaction and function partition), the function list should be created with main ideas and short description.



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

3.2. FUNCTION LIST.

- Reasons for function list creation:
  - Brainstorming to develop necessary sub functions from big functions.
  - Making combination of functions in the design from function list.
  - Showing the ideas of designers for how to implement or what implementations are chosen from various methods.
- Benefits from function list creation:
  - Generating sufficient verification items.
  - Avoiding lacking verification items.
  - Understanding of other designers from top functions to bottom functions.
  - Generating combination cases for verification and design.

3.3. FUNCTION INTERACTION.

- Function interaction.
  - Function interaction is generated from function list.
  - Function interaction shows the relationships between functions in function list.
- Reason for function interaction making.
  - Showing available and unavailable combination functions.
  - Avoiding unnecessary verification items for unreachable combinations.
  - Creating interface specification for related functions.
  - Reducing unnecessary logic design for unreachable combinations.
- Risk of lacking function interaction.
  - Lacking complex combination items between functions.
  - Understanding of functional relationships for other designers to reuse the design.
  - Avoiding lacks in consideration for complex functions during design phase.

3.3. FUNCTION INTERACTION.

- Function interaction – matrix table.

	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						

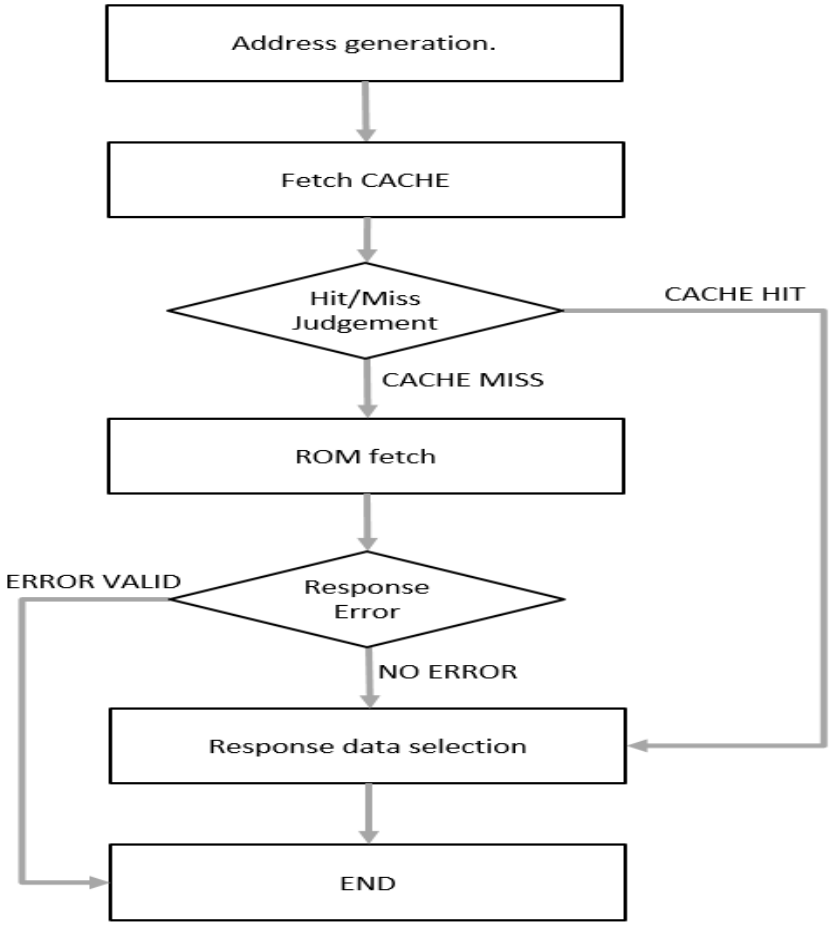
o: these functions have relationships together.

x: there is no relationship between functions.

- Show relationships between functions in the design.
- In case of complex design which needs more than 2 combinations, designers can try to make function partition (be explained later) first to reduce the complex of design before making matrix table.

3.3. FUNCTION INTERACTION.

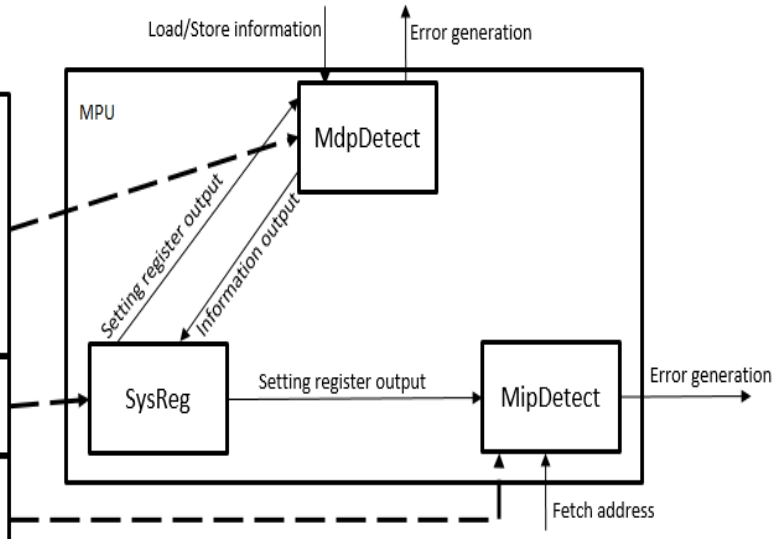
- Function interaction – flow chart.
- Show the timing relationship or sequences between functions.
- Usually apply for FSM or designs that consist of various phases to handle input.



3.4. FUNCTION PARTITION.

- Function partition.
  - Function partition is also created from function list.
  - Functions in function list are grouped into physical blocks.
  - All functions in function list must be located in specific blocks.
  - Ex: MdpDetect module consists of “Load/Store violation checking” and “Self-check function” functions.

Memory protection function. MPU module.	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.	



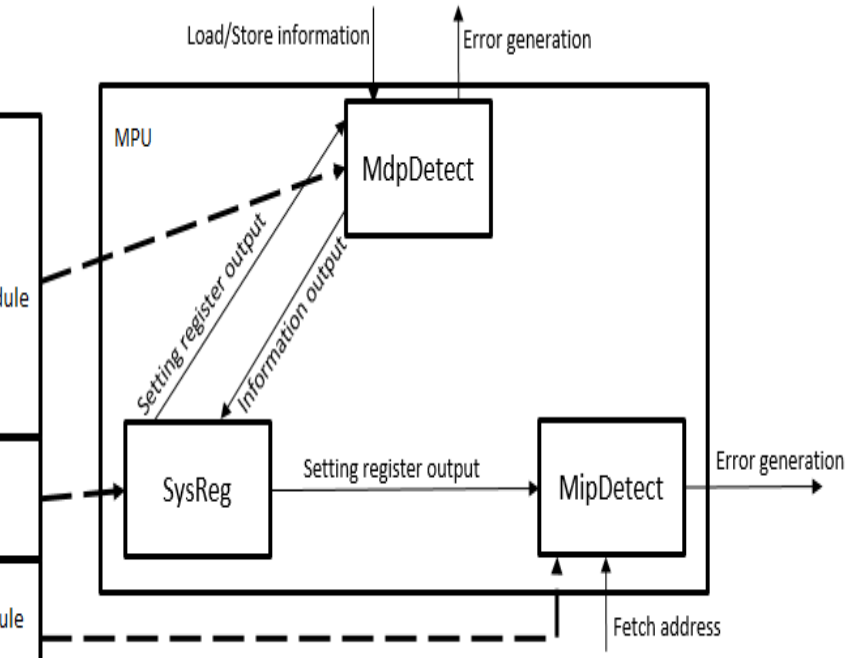
3.4. FUNCTION PARTITION.

- Reason for function partition making.
  - Confirming the equivalence between function list and functions in submodule. If not equivalence, designers have to consider the sufficient functions in the design.
  - Creating physical block diagram.
  - Considering the effectiveness of physical block division at early phase.
  - Enhancing other designers' understanding for relations between submodules.
- Problems of lacking function partition.
  - Other designers and reviewers cannot consider the effectiveness of physical block division during design review.
  - Other designers may feel difficult to understand the relationships between submodules.
  - Designers may lack to implement some functions during design phase.

4. DESIGN DOCUMENT – INTERFACE SPECIFICATION.

- Interface specification shows the connections and relationships between modules or submodules.
- Interface specification can be extracted from function interaction and function partition.

Memory protection function. MPU module.	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.	



4. DESIGN DOCUMENT – INTERFACE SPECIFICATION.

- Interface specification contents.
  - Interface list: list of interface signals, bit width, reset value, and description for the interface.
  - Interface explanation: explain the meaning of signals in interface list, the relationships of signals, or protocol requirement.
- Reasons for interface specification making.
  - Show the protocol between modules.
  - Show the requirement between modules for input or output data.
  - Show the relationships between signals such as control signal and data signal.
  - Confirm necessary input to support functions in the design.
- Risks of lacking interface specification.
  - Mismatch in operation between modules, especially hand-shake signal.
  - Mismatch in timing for when data should be valid or available.
  - Lack important signals to develop functions → re-design → increase workload.

4.1. INTERFACE LIST.

- Interface list is a list of signals which can be written into a table.
- The table shows the status of each signal such as input, output, reset value, or short description.
- The interface list should be grouped based on functions. For example, one section is used for interface of function A.1, and another section is used for interface of function A.2.
- The interface list format may be varied in designer leaders or customer requests.
- These information below is the minimum requirement for interface list.
  - List of signal.
  - Input/output port.
  - Bit width.
  - Initial value after reset.
  - Short description.
- The example below is a suggestion from author for interface list making.

4.1. INTERFACE LIST.

- Example for fetch request between ICC and ROM memory.

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.

4.1. INTERFACE LIST.

- Signal: describe the signal name as well as bit width.
- In/Out: specify the signal is output port or input port.
- Initial: it means the value of interface after reset. The meaning of the characters in the column are:
  - Exact value (1'b0, 1'b1, or defined value): after reset, the port must be that value.
  - Undefined: the value can be any value. It is usually used for data signal which need control signal to valid the data.
  - D.C: mean "don't care". After reset, the value of this port can be any value and it would no affect the operation of system.
  - Optional: it is usually used for pin setting signal. It means the value can be free, yet its value can affect the operation of system. The possible value should be clearly defined for this kind of interface.
- Description: it shows the short description for the function of each signal.

4.2. INTERFACE EXPLANATION.

- This part explains the meaning, the relationship, or the protocol of interface signals.
- The information helps:
  - To clearly understand the protocol.
  - To easily judge whether the timing implementation is sufficient.
  - To confirm the availability of data between modules.
  - To sufficiently implement the functions in each modules based on the data flow.
  - To check performance implementation sooner such as bus access latency.
  - To apply in interface verification activities.

4.2. INTERFACE EXPLANATION.

- Example for interface explanation based on interface list above.
- The explanation below shows the relationship between control signals and data signals.
- It shows the status of control signals and valid signals for data signals.

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.

4.2. INTERFACE EXPLANATION.

- The explanation below shows the protocol between modules.
- The protocol is extremely important from the viewpoint of module communication.
- If not clearly define, the bugs would appear in hand-shake signals.

- *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.*

4.2. INTERFACE EXPLANATION.

- The explanation for protocol is extremely important as the problems for communication may occurs.
- In some cases, the additional timing charts or examples for protocol should be included.

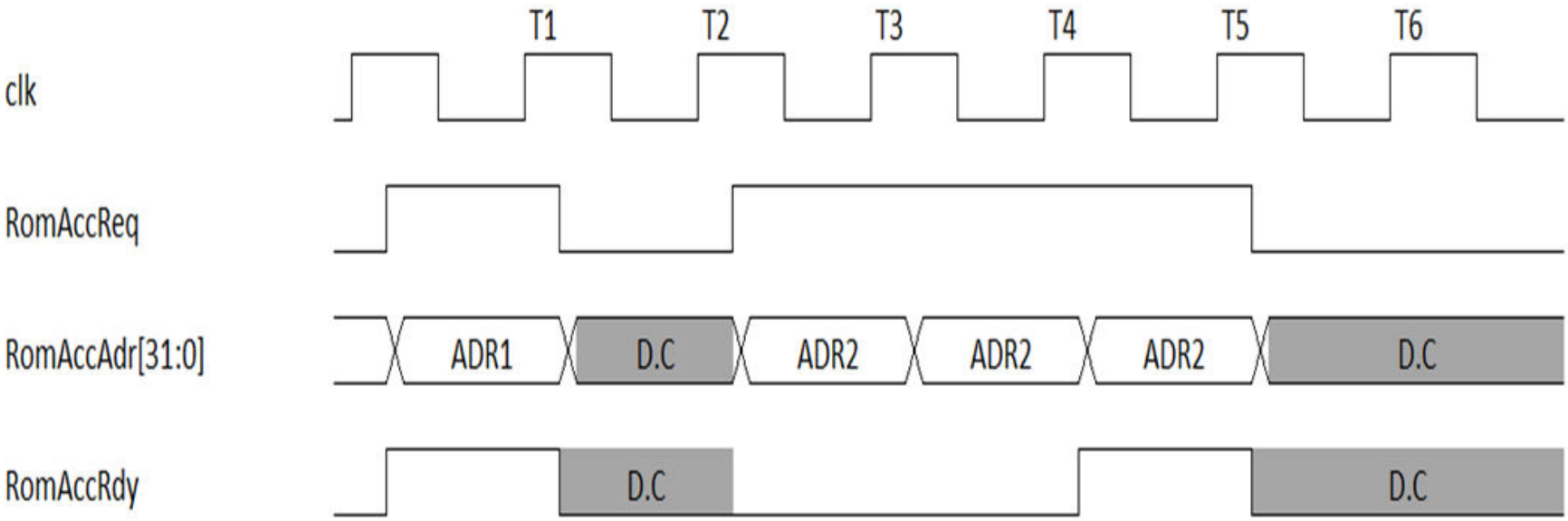


Figure <number>: the example for ICC and ROM memory protocol.

4.2. INTERFACE EXPLANATION.

- The explanation for timing chart should be included.
- It helps readers to understand the meanings, reasons, key points, or important/special points in timing charts.

- 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.

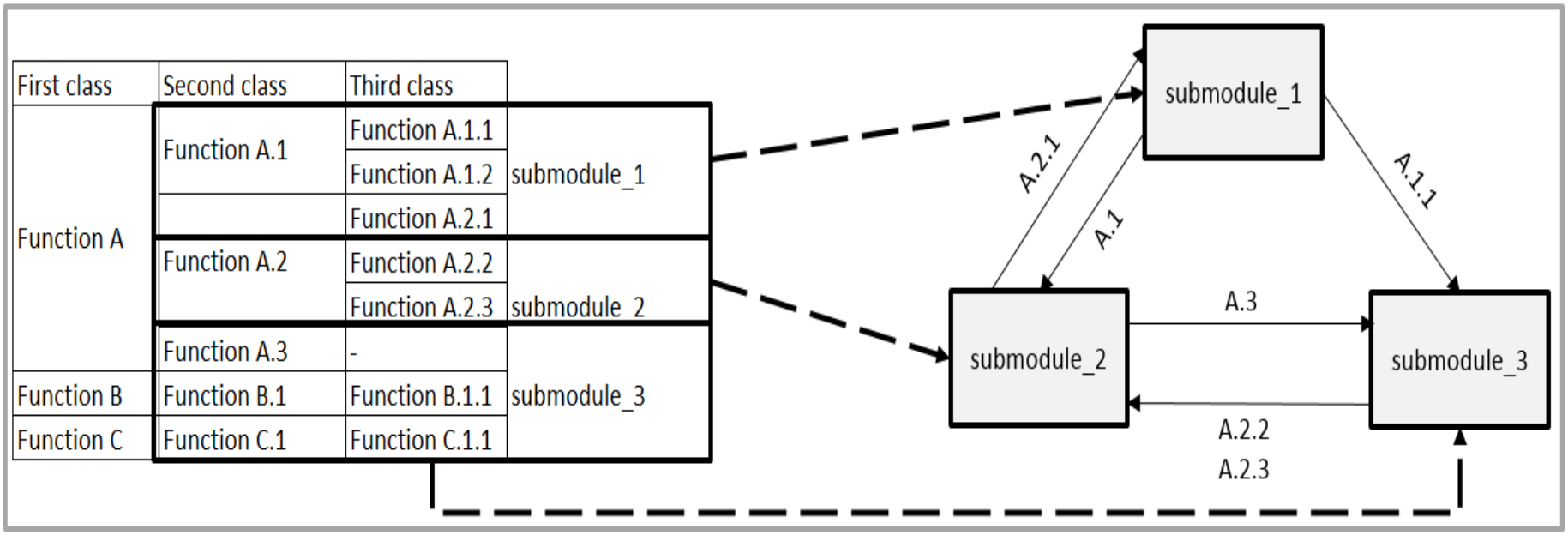
- From T3 to T4, the RomAccReq is valid, yet ROM memory is busy. Hence, RomAccRdy is 1'b0 to show the status of ROM memory. During the pending phase, the request and the address from ICC must be kept stable as the requirement of protocol.

- T5: RomAccRdy is asserted to indicate the availability to receive request from ICC.

- T6: RomAccReq is cleared, and the operation is same as T2.

5. DESIGN DOCUMENT – IMPLEMENTATION SPECIFICATION.

- Implementation specification is also created from function interaction and function partition.
- Block diagram and submodules are generated from function partition.
- Implementation specification try to explain the IDEA for how to support functions.



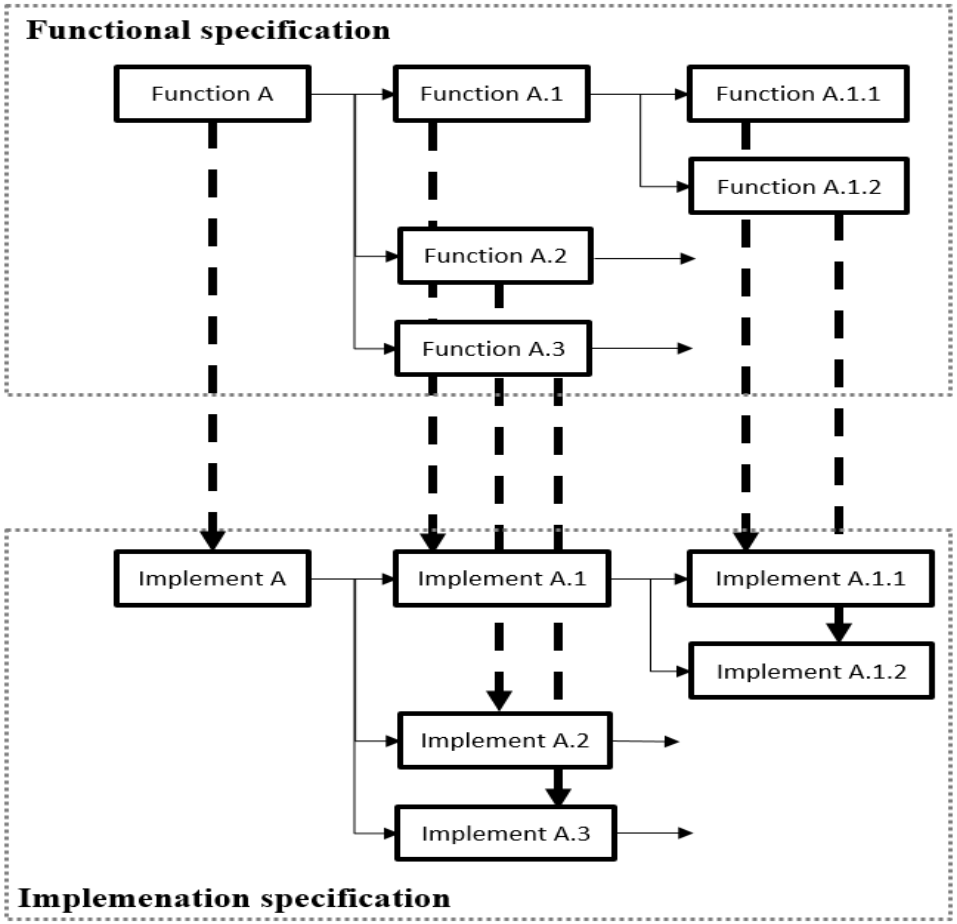
First class	Second class	Third class	
Function A	Function A.1	Function A.1.1	submodule_1
		Function A.1.2	
		Function A.2.1	
	Function A.2	Function A.2.2	submodule_2
		Function A.2.3	
	Function A.3	-	submodule_3
Function B	Function B.1	Function B.1.1	
Function C	Function C.1	Function C.1.1	

5. DESIGN DOCUMENT – IMPLEMENTATION SPECIFICATION.

- What should be written into implementation specification?
  - Show the IDEAS of designers for how to implement a function rather than complex combination logic.
  - Explain how to implement a function by using:
    - ✓ Block diagram.
    - ✓ True table.
    - ✓ Timing chart.
    - ✓ State machine.
  - Show special logic which is used to:
    - ✓ Fixing bug.
    - ✓ Improving timing violation.
    - ✓ Improving power consumption.

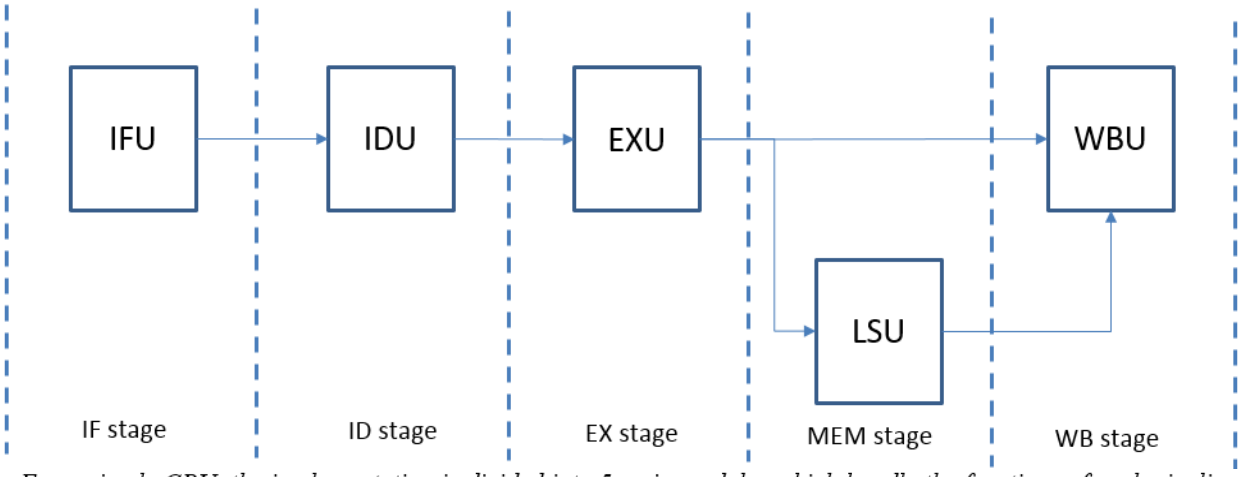
5. DESIGN DOCUMENT – IMPLEMENTATION SPECIFICATION.

- How to develop implementation specification?
  - The implementation should be developed from top level to bottom one.
  - Generally, each functions should be explained by corresponding implementation specification.



5.1. BLOCK DIAGRAM AND HIERARCHY.

- Block diagram is generated from function partition.
- Block diagram shows the overall and position of submodules in the design.
- Block diagram helps readers to understand the structure of the design.
- In some cases, block diagram is described as data flow for designs consisting of various timing phases.
- It is recommended to put the explanation for the main functions of each submodule.



- 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.

5.1. BLOCK DIAGRAM AND HIERARCHY.

- Hierarchy shows submodule's name, main functions, and the layers of each submodules.
- It is also created from function partition.

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

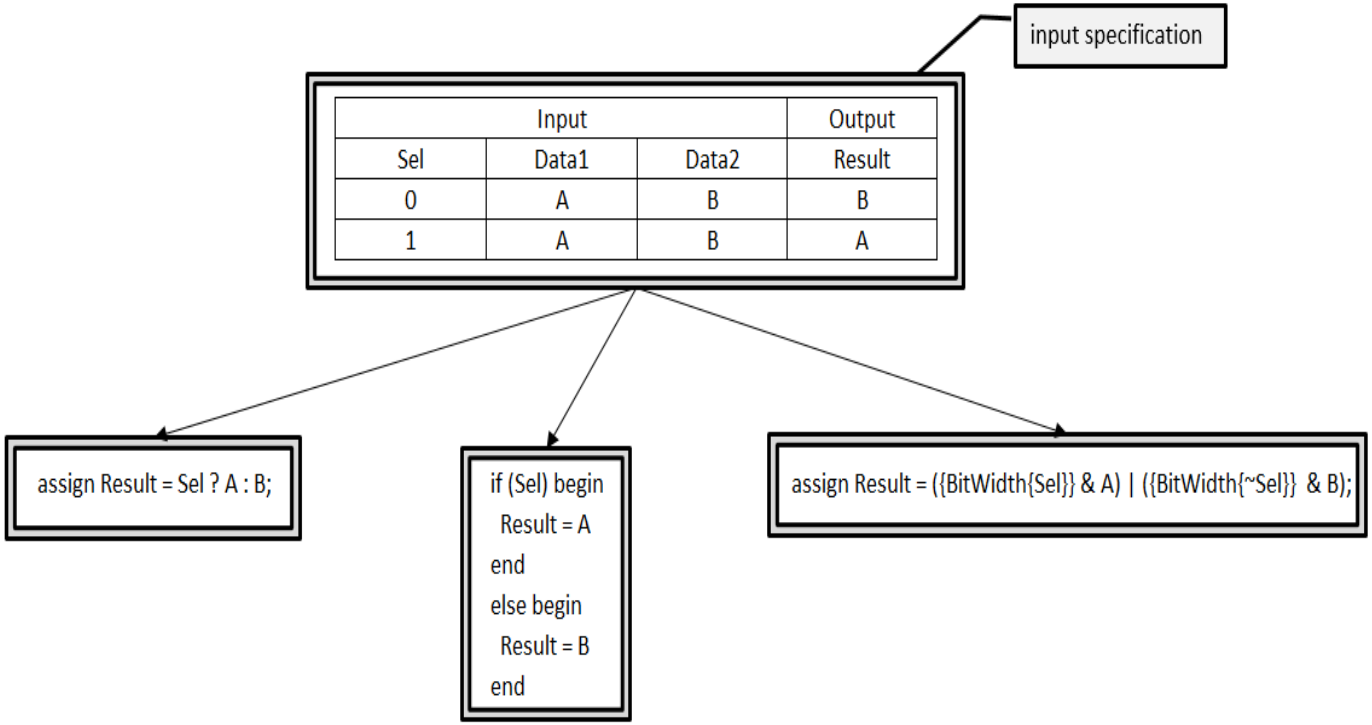
5.2. TRUE TABLE.

- True table shows the result of a function based on specific input.
- True table consists of two main parts:
  - Input is specific conditions which are issued to the function from other logic.
  - Output is the result corresponding with each input.
- In some cases, the description is needed to explain each parts in true table.
- True table helps:
  - Designers to consider possible cases of the design.
  - Verification members create sufficient verification items.

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

5.2. TRUE TABLE.

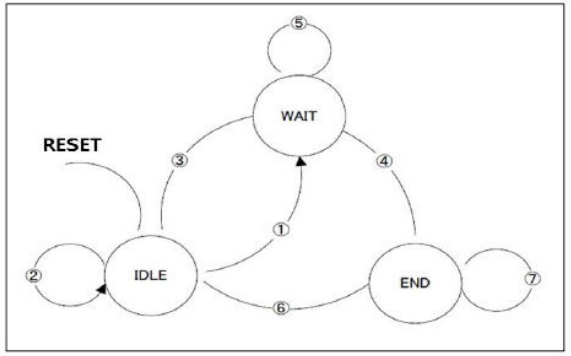
- In some cases, true table is much more important than complex logic picture.
- From a request, designers may have various ways to develop. Hence, complex logic picture makes the design document complicated and difficult to read, understand, or review.



5.3. STATE MACHINE.

- State machine is used to explain functions which consist of various timing phases.
- State machine can be split into two parts:
  - ➢ State meaning: explain the meaning of each states.
  - ➢ State transaction: explain the changing conditions, destination, and description.
- Generally, state machine is used for functions with various timing phases, so timing chart should be added to clearly explain state machine.

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

5.4. TIMING CHART.

- The timing chart is used to explain the timing in RTL design, the constraints, or relationships between signals.
- Timing chart also describes the state and the operation of FSM.
- Timing chart making can be created as timing chart in interface explanation.

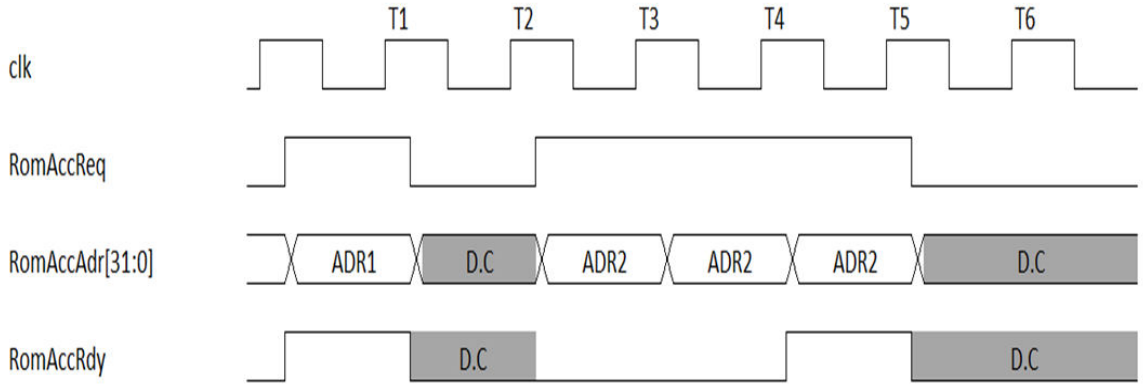


Figure <number>: the example for ICC and ROM memory protocol.

- 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.

6. DESIGN DOCUMENT – SUMMARY.

- Although design document making costs time, it plays an important role in the success of the project.
- Design document format bases on the style of design leaders or request of customers.
- The principle requirements for design document are:
  - Precise: information must be clearly and correctly written.
  - Concise: the idea should be written for other to easily understand.
  - Clearness: information in document must be understood as a same way for anyone.
- One tip for document making is: always try to explain the meaning, the purpose of timing chart/true table/etc., or the design reason.
- Another point is how to express idea in English. There are two reference document that may help to write design document, to a certain extent.
  - HowToWriteDocument.docx – Appendix: Building a paragraph.
  - How to write an effective business document – DungVuu <dung.vuu.xa@rvc.renesas.com>

7. HISTORY.

Date	Author	Part	Description
2017/12/12	Vu Huynh	-	New creation
2018/01/31	Vu Huynh	0. Overall	Add overall content in the slide.
		1. Design document – introduction.	Add definition for design document.
		2. Design document – Partition.	Change order of interface and implementation specification.
		2. Design document – Partition.	Remove Figure at slide 14 to avoid confusing.
		3.3. Function interaction. 3.4. Function Partition. 4. Design document – interface specification.	Change the example as real design.

BIG IDEAS FOR EVERY SPACE

THANK YOU FOR YOUR ATTENTION!