Testing

07 February 2021 04:06 PM

1) What is MIL and SIL ?
    - When we develop the Controller model and verify if the Controller can control the Plant as per the requirement. This step is called Model-in-Loop (MIL) and we are testing the Controller logic on the simulated model of the plant. If our Controller works as desired, we should record the input and output of the Controller which will be used in the later stage of verification
    - Once your model is verified (i.e., MIL in the previous step is successful), the next stage is Software-in-Loop(SIL), where we generate code only from the Controller model and replace the Controller block with this code. Then run the simulation with the Controller block (which contains the C code) and the Plant, which is still the software model (similar to the first step). This step will give us an idea of whether our control logic i.e., the Controller model can be converted to code and if it is hardware implementable. we should log the input-output here and match with what we have achieved in the previous step. If we experience a huge difference in them, we may have to go back to MIL and make necessary changes and then repeat step MIL and SIL again. If we have a model which has been tested for SIL and the performance is acceptable we can move forward to the next step.

2) What is Overflow errors and Underflow ?
    - An overflow error indicates that software attempted to write data beyond the limits of memory.
    - Underflow is a condition which occurs in a computer or similar device when a mathematical operation results in a number which is smaller than what the device is capable of storing. It is the opposite of overflow, which relates to a mathematical operation resulting in a number which is bigger than what the machine can store. Similar to overflow, underflow can cause significant errors.

3) What are the constrain while writing the test cases ?
    - Write test cases priority wise according to your model
    - Write easy and understandable test cases
    - Always keep in mind the end user requirement
    - Monitor test cases regularly until your project get completed
    - Write unique test cases and avoid duplication of test cases
    - Apply the 80/20 rule while writing test cases that is 20% test cases will satisfy the 80% of model coverage of your model

4) What is logging of signal ?
    - Signal logging acquires signal data during run time and store it in target computer after completion of simulation, we can view this data into data inspector
    - We can compare simulation result data of two or more signal

5) What are the analysis and debugging of your model ?

- Analysis of model covers the following points.
    - Model advisor (MAAB)
    - Test manager (MIL & SIL)
    - Design verifier (Check Capability)
    - Coverage analysis (DC, CC, MCDC Coverage)
    - Data type decision (Fixed point tool, Single Precision conversion)
- Debugging of simulink model
    - Right click on any signal where you want to add the breakpoint
    - Select the option "Add Conditional Breakpoint"
    - Run the simulation step wise
    - Whenever condition get true at the applied breakpoint then the green symbol will appears on the applied breakpoints.

4) What is overflow ?
   - When you want to save a number in variable but the variable size not enough to store that number then overflow is happen eg. If we want to store 200 in int8 but int8 can store up to 127 after you store overflow will happen.

5) What is data dictionary ?
   - A data dictionary is a persistent repository of data that are relevant to your model. You can also use the base workspace to store design data that are used by your model during simulation. However, a data dictionary provides more capabilities.
   - The dictionary stores design data, which define parameters and signals, and include data that define the behaviour of the model. The dictionary does not store simulation data, which are inputs or outputs of model simulation that enter and exit Inport and Outport blocks.
   - Advantages
     - We can edit our variable and can change its data and data type from data dictionary
     - We don't need to load parameter file every time when we open model variable automatically load when we open model
     - It provide more capabilities -
       - Track changes
       - Control access
       - Partition data
       - Share data
   - Single data dictionary
     - Benefits
       - All data in one place
       - Shared data centralized management
     - Use case
       - Single developer
       - Small design
   - Shared data dictionary
     - Benefits
       - Concurrent development
       - Clear data ownership
     - Use case
       - Large teams
       - Complex designs
   - Data dictionary save data in form of .sldd file

6) What is data inspector ?
   - Data inspector is a window where we can able to see logged signals from your model
   - Logged signals shows the output in a graphical window(view) in data inspector
   - You can be able to compare the two signals for the analysis purpose compare option provides the baseline and compared to option for the comparison
   - We can export data in mat file or base workspace

7) What you can do using model explorer ?
   - To view, modify and add the elements in the simulink model, stateflow chart and workspaces then model explorer is used.
   - It lets you focus on specific elements without navigating through the model or chart you can combine search criteria and interactivity define the result.

8) How to link requirement to your simulink model ?
    - Create a model and go to the analysis
    - In analysis go to requirements then requirements editor
    - In new window add new requirement set which is nothing but system level or high level requirement
    - Click on add requirement which will be added in the system level requirement these are software requirement
    - You can add the customer id summary and description of the parent system and software requirements
    - To add the child requirements right click on the parent requirement and select add the child requirements.
    - Do this for the other child requirement
    - After this go to req. perspective window either go-to from right button corner or go-to from analysis --> req. --> perspective window
    - To link the req. to particular block or the port drag the respective req. to that block or port
    - Otherwise to link the req. to that block or port.

9) Why do we need SIL even if we test for MIL ?
    - Mil is the procedure where we are comparing the simulation output with expected output but in the real time we are using the generated code which is we need to deploy in the hardware system. So before deploying we must have to check the generated code therefor SIL is perform even after MIL testing is done on our model.
    - After generate code some logic are not execute in that code that is dead logic so when we perform MIL we can remove dead logic easily which we can not remove during SIL.

10) What is integrated testing and system testing ?
    - Integrated testing :-
        - Integrated testing is nothing but the combining all the unit model and then performing the testing. In short integrated system is nothing but summation of all models and then testing is done.
    - System testing :-
        - System testing is nothing but the testing of the entire system which contains unit model, integrated model, controller testing and hardware testing. if perform integration system.

11) Difference between unit testing and MIL testing ?
    - Unit testing is nothing but the testing of only single model developed by the respective developer and then we perform MIL and SIL on that model only
    - MIL testing is nothing but integrated testing which contain all the unit model which are generated together and then testing applied on it.

12) Which tool you use for test case generation ?
    - Design verifier
    - Signal builder

13) What are the bugs in MIL Testing ?
    - Mismatched data types of signals
    - Algebraic loop error
    - Deflection in output i.e. (simulation output and expected output)
    - Coverage may differ (depend on test cases)
    - Undefined variable

14) What are the bugs in SIL testing ?
    - Compiler error (if its not installed)
    - System must be atomic
    - Cant apply SIL for virtual subsystem or blocks

- Target must be set to ert.tlc for embedded coder
- Configuration parameter of model and harness model must be same

15) Can we generate code without using atomic subsystem ?
    - Yes we can generate c code without using atomic subsystem

16) What is variant subsystem ?
    - A Variant Subsystem block contains two or more child subsystems where one child is active during model execution.
    - The active child subsystem is referred to as the active variant.
    - You can programmatically switch the active variant of the Variant Subsystem block by changing values of variables in the base workspace, or by manually overriding variant selection using the Variant Subsystem block dialog.
    - The active variant is programmatically wired to the Inport and Outport blocks of the Variant Subsystem by Simulink during model compilation.
    - Command to activate variant subsystem
        - Add = Simulink.Variant('ADD == 1')
        - Sub = Simulink.Variant('ADD == 0')

17) Difference between unit testing and function testing.

Unit testing	Function testing
• Testing unit in isolation	• Testing the function as per user requirement
• Easy to write and execute	• More complex compared to unit testing
• Its a white box testing	• It's a black box testing
• Its test individual modules or units	• It test entire application functionality
• Ideally written from starting of development	• When features has been built
• Written by developers	• Written by testers

18) Difference between static testing and dynamic testing.

Static testing	Dynamic testing
• It is about prevention of defects	• It is about finding and fixing of defects
• It check the model requirement documents to find errors	• It checks the functional behaviour of software system
• It does verification process	• It does the validation process
• Performed before compilation process	• Performed after compilation process

19) MISRA C Guidelines
    - The Motor Industry Software Reliability Association (MISRA®[1]) has established guidelines for the use of the C Language in Critical Systems (MISRA C®).
    - Check for blocks not recommended for c/c++ production code deployment eg. Derivative, integrator, PID, display, scope, strop simulation.
    - Check for unsupported block names
    - Check for usage of assignment blocks
    - Check for switch case expression without a default case
    - Check for bitwise operations on signed integer

- Check integer word length
- Check bus object names that are used as bus element names

20) Storage classes of MATLAB.

Storage Class	Description
• 'ExportedGlobal'	• Defines the variable in the Variable Definitions section of the C file entry_point_name.c. • Declares the variable as an extern in the Variable Declarations section of the header file entry_point_name.h • Initializes the variable in the function entry_point_name_initialize.h.
• 'ExportedDefine'	• Declares the variable with a #define directive in the Exported data define section of the header file entry_point_name.h.
• 'ImportedExtern'	• Declares the variable as an extern in the Variable Declarations section of the header file entry_point_name_data.h. The external code must supply the variable definition.
• 'ImportedExtern Pointer'	• Declares the variable as an extern pointer in the Variable Declarations section of the header file entry_point_name_data.h. The external code must define a valid pointer variable.

21) UAT - User acceptance testing ?
- User Acceptance Testing (UAT) is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment.

22) Integration testing ?
- Integration testing is a level of software testing where individual units / components are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.

23) Regression testing ?
- Regression testing is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features.
- Regression Testing is nothing but a full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.
- This testing is done to make sure that new code changes should not have side effects on the existing functionalities. It ensures that the old code still works once the latest code changes are done.

24) Scrum
- Scrum is a framework that helps teams work together. Much like a rugby team (where it gets its name) training for the big game, Scrum encourages teams to learn through experiences, self-organize while working on a problem, and reflect on their wins and losses to continuously improve.
- While the Scrum I'm talking about is most frequently used by software development teams, its principles and lessons can be applied to all kinds of teamwork. This is one of the reasons Scrum is so popular. Often thought of as an agile project management framework, Scrum describes a set of meetings, tools, and roles that work in concert to help teams structure and manage their work

25) What is Plant model ?
- Plant model is a actual replica of real world prototype
- For eg. My project is a controller model which I developed to control the climate of car to maintain the ambient temperature and the actual car model is a plant model.

26) What is MIL ?

- MIL is a model in loop which comes in the verification part of model based design approach.
- First we have to develop a model of the actual plant in matlab simulink environment which capture most of the important future of the system.
- After the plant model is created we will develop the controller model and verify if the controller can control the plant as per requirement.
- This step is model in loop(MIL) and you are testing the controller logic on the simulated model of the plant.

27) What is SIL ?

- Once your model is verified(ie. MIL is successful) the next stage is software in loop(SIL). where you generate code only from the controller model and replace the controller block with this code.
- Then run the simulation with the controller block(which contain c code) and the plant model.
- This step will give you an idea of whether your control logic ie the controller model can be converted to code and if it is hardware implementable.
- You should log the input output here and match with what you have achieved in the MIL.
- If you experience a huge difference in them you may have to go back to the MIL and make necessary changes and then repeat MIL and SIL

28) What is PIL ?

- After SIL the next step is PIL(Processor in Loop)
- In PIL we will put the controller model cod onto an embedded processor and run a close loop simulation with the simulated plant model.
- This step will help you identify if the processor is capable of running the developed control logic
- Check your PIL output with expected output which you have get from MIL and SIL
- If your code is not perform well in the PIL then go back to your code and modify and do MIL and SIL again and then letter do PIL again.

29) Execution Coverage (EC)

- Execution coverage is the most basic form of coverage. For each item, execution coverage determines whether the item is executed during simulation.

30) Decision Coverage (DC)

- Decision coverage analyses elements that represent decision points in a model, such as a Switch block or Stateflow® states. For each item, decision coverage determines the percentage of the total number of simulation paths through the item that the simulation traversed.

31) Condition Coverage (CC)

- Condition coverage analyses blocks that output the logical combination of their inputs (for example, the Logical Operator block) and Stateflow transitions.
- A test case achieves full coverage when it causes each input to each instance of a logic block in the model and each condition on a transition to be true at least once during the simulation, and false at least once during the simulation.
- Condition coverage analysis reports whether the test case fully covered the block for each block in the model.

32) Modified Condition/Decision Coverage (MCDC)

- Modified condition/decision coverage analysis by the Simulink Coverage software extends the decision and condition coverage capabilities. It analyses blocks that output the logical combination of their inputs and Stateflow transitions to determine the extent to which the test case tests the independence of logical block inputs and transition conditions.
  - A test case achieves full coverage for a block when a change in one input, independent of any other inputs, causes a change in the block output.

        - A test case achieves full coverage for a Stateflow transition when there is at least one time when a change in the condition triggers the transition for each condition.
    - If your model contains blocks that define expressions that have different types of logical operators and more than 12 conditions, the software cannot record MCDC coverage.
    - Because the Simulink Coverage MCDC coverage may not achieve full decision or condition coverage, you can achieve 100% MCDC coverage without achieving 100% decision coverage.
    - Some Simulink objects support MCDC coverage, some objects support only condition coverage, and some objects support only decision coverage.

33) Why coverage ?
    - To checks all blocks being executed
    - To check whether all decision and condition are being satisfied.
    - To check weather the model is ready according to the client requirements.

34) Types of model coverages.
    - Block execution
        - Check execution coverages of every blocks
    - Decision coverages
        - Checks decision points in model eg. Switch or stateflow
    - Condition coverage
        - Check output of logical combination and stateflow transitions.
    - MCDC
        - Decision and condition coverage
    - Colour
        - Green --> Full Coverage
        - Red --> Partial Coverage
        - Grey --> Filtered Coverage
    - Current run
        - Shows coverage of current simulation
    - Delta
        - Shows difference between cumulative and current run
    - Cumulative
        - Shows coverage of all cumulative runs

35) What is verification and Validation of your project ?
    - Verification
        - Model verification includes checking against standards, checking for design errors, proving properties and running simulation to demonstrate that result match expected output.
        - Coverage measurement is also part of model verification same process applies for generated code.
    - Validation
        - We can validate requirement by executing our system prototype connected to User interface or plant model

36) Different Testing Methods.
    - Positive Testing
        - Positive Testing is a type of testing which is performed on a software application by providing the valid data sets as an input.
        - It checks whether the software application behaves as expected with positive inputs or not.
        - Positive testing is performed in order to check whether the software application does exactly what it is expected to do.
    - Negative Testing
        - Negative Testing is a testing method performed on the software application by providing invalid or

improper data sets as input.
    - It checks whether the software application behaves as expected with the negative or unwanted user inputs. The purpose of negative testing is to ensure that the software application does not crash and remains stable with invalid data inputs.
- Component Testing
    - Component testing is defined as a software testing type, in which the testing is performed on each individual component separately without integrating with other components.
    - It's also referred to as Module Testing when it is viewed from an architecture perspective. Component Testing is also referred to as Unit Testing, Program Testing or Module Testing.
- Smoke Testing
    - Smoke Testing is a software testing process that determines whether the deployed software build is stable or not. Smoke testing is a confirmation for QA team to proceed with further software testing.
    - it consists of a minimal set of tests run on each build to test software functionalities. Smoke testing is also known as "Build Verification Testing" or “Confidence Testing.”
      In simple terms, we are verifying whether the important features are working and there are no showstoppers in the build that is under testing.
    - It is a mini and rapid regression test of major functionality. It is a simple test that shows the product is ready for testing. This helps determine if the build is flawed as to make any further testing a waste of time and resources.
- Sanity Testing
    - Sanity testing is a kind of Software Testing performed after receiving a software build, with minor changes in code, or functionality, to ascertain that the bugs have been fixed and no further issues are introduced due to these changes. The goal is to determine that the proposed functionality works roughly as expected. If sanity test fails, the build is rejected to save the time and costs involved in a more rigorous testing.
    - The objective is "not" to verify thoroughly the new functionality but to determine that the developer has applied some rationality (sanity) while producing the software. For instance, if your scientific calculator gives the result of 2 + 2 =5! Then, there is no point testing the advanced functionalities like sin 30 + cos 50.

Smoke Testing	**Sanity Testing**
Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine	Sanity Testing is done to check the new functionality/bugs have been fixed
The objective of this testing is to verify the "stability" of the system in order to proceed with more rigorous testing	The objective of the testing is to verify the "rationality" of the system in order to proceed with more rigorous testing
This testing is performed by the developers or testers	Sanity testing in software testing is usually performed by testers
Smoke testing is usually documented or scripted	Sanity testing is usually not documented and is unscripted
Smoke testing is a subset of Acceptance testing	Sanity testing is a subset of Regression Testing
Smoke testing exercises the entire system from end to end	Sanity testing exercises only the particular component of the entire system
Smoke testing is like General Health Check Up	Sanity Testing is like specialized health check up

- System Testing
    - System testing is a level of testing that validates the complete and fully integrated software

product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system.

- Ultimately, the software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

- Regression Testing
    - REGRESSION TESTING is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features.
    - Regression Testing is nothing but a full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.
    - This testing is done to make sure that new code changes should not have side effects on the existing functionalities. It ensures that the old code still works once the latest code changes are done.

37) Black Box and White Box Testing.

Black Box Testing	White Box Testing
• It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it.	• It is a way of testing the software in which the tester has knowledge about the internal structure or the code or the program of the software.
• It is mostly done by software testers.	• It is mostly done by software developers.
• No knowledge of implementation is needed.	• Knowledge of implementation is required.
• It can be referred as outer or external software testing.	• It is the inner or the internal software testing.
• It is functional test of the software.	• It is structural test of the software.
• This testing can be initiated on the basis of requirement specifications document.	• This type of testing of software is started after detail design document.
• No knowledge of programming is required.	• It is mandatory to have knowledge of programming.
• It is the behaviour testing of the software.	• It is the logic testing of the software.
• It is applicable to the higher levels of testing of software.	• It is generally applicable to the lower levels of software testing.
• It is also called closed testing.	• It is also called as clear box testing.
• It is least time consuming.	• It is most time consuming.
• It is not suitable or preferred for algorithm testing.	• It is suitable for algorithm testing.
• Can be done by trial and error ways and methods.	• Data domains along with inner or internal boundaries can be better tested.
• **Example:** search something on google by using keywords	• **Example:** by input to check and verify loops

38) Boundary condition Testing

- Boundary testing is the process of testing between extreme ends or boundaries between partitions of the input values.
- So these extreme ends like Start- End, Lower- Upper, Maximum-Minimum, Just Inside-Just Outside values are called boundary values and the testing is called "boundary testing".