1. **Guidelines to follow using Merge Block?**
    - Always use conditionally executed subsystems to drive Merge blocks.
    - Ensure that all inputs signals have the same data type.
    - Ensure that all input signals have the same sample time.
    - set the Output when disabled parameter to held.
    - If the output of a Model block is coming from a MATLAB Function block or a Stateflow® chart, do not connect that output port to the input port of the Merge block.
    - Don't branch a signal that inputs to a merge block.
    - Ensure only one subsys. is active at a time.

2. **Any 5 debug processes:**
    - With the help of scope.
    - Enable stepping back options
    - display value label of the selected port.
    - Using breakpoints.
    - In view option symbols shows.
3. **Difference between**

Fixed Step Solver	**Variable Step Solver**
Fixed-step solvers solve the model at regular time intervals from the beginning to the end of the simulation.	Variable-step solvers vary the step size during the simulation.
The step size remains constant throughout the simulation.	A variable-step solver increases or reduces the step size to meet the error tolerances that you specify.
It is easy to use in real-time application.	Difficult to use in real-time.
The step size is fixed it means we can decide the step size.	In VSS, the system takes step size as auto,so it auto calculates and it depends on the model dynamics.
It does not detect zero-crossing.	It capture zero-crossing

Unit Delay	**Memory Block**
Unit delay block holds and delays its input by the sample period we specify.	Memory block holds and delays its input by one major integration time step.
Each signal can be scalar or vector.	Each signal can be scalar, vector, matrix or N-D array.
Unit delay in discrete model.	Memory block use in simscape.
It has sample time as -1.	It does not have a sample time.
It takes less memory compared to memory block.	It takes more memory compared to memory block.

Bus Creator	Mux
It takes more memory.	It takes less memory than bus creator.
It works like structure.	It works like array.
Different types of data signals.	Similar types of data signals.
Bus creator block combines a set of input elements into a bus.	Mux block combines its inputs into a single vector output.
It provides bus signals.	It provides vector signals.

4. **Why not use memory block instead of delay block?**

Memory Block use in Simscape or Continuous changes model, Unit delay in Discrete Model.

Enabled Subsystem	Triggered subsystem
Enable control signal has a positive value.	TS signal can have a + ve & - ve value.
Enable control signal rises through zero.	Triggered control signal rises or fall through zero.
It has 2 conditions. (properties) 1.Held 2. Reset	It has 4 conditions: 1.Rising 2. Falling 3. Either 4. Function-call

Simulink	Stateflow
Step execution not depend on each other.	Step execution depends on each other.
You cannot control event in Simulink.	You can control event in stateflow.
At one sample time whole model can run.	At 1 sample time, only one state one action can be active.
Simulink model after coding usually difficult.	Stateflow model after coding usually easy as compared to Simulink.
Debugging is easy.	Debugging is not easy.
Readability of code is not easy.	Readability of code is easy.
We can not check or execute conditional logic easily.	We can check or execute conditional logic easily.

Source	Sink
Source blocks generally has the input blocks which provides input to the systems.	Sink blocks are output collecting blocks.
Signal builder, constant, repeating sequence stair, from workspace, input.	To workspace, scope, outport.

Switch block	Multiport Switch
It has 2 data input and 1 control unit.	It has number of data inputs.
True and false inputs are there.	It can be : zero based, one based, indices we specify.

Exclusive State	Parallel State
Substates with solid borders indicate exclu. State decomposition.	Substates with dotted lines or dashed boders indicates parallel state decomposition.
Only one substate can be active at a time when it's a excl. decom.	All substates can be active at a time when its parallel state decom.
It can indicate with (OR).	It can indicate with (And).
We can give default transition in Exclu.	No need to give default transition in parallel.

Atomic	Non-Atomic
It has a Dark-line border.	It has Thin-line border.
You can't expand subsystem	You can expand subsystem
At the time of execution order it is considered as separate entity.	At the time of execution order it is not considered as separate entity.it considered at same level.
Logic inside the subsystem executed completely first then only execution come out of the system.	It doesn't required logic inside the subsystem should execute first.
You can control Function Packaging.	You can't control Function packaging.
These are non-virtual subsys.	These are virtual subsys.

History Junction	Junction
A history junction records the activity of substates inside superstates.	Junction divides the transition into transition segment.
We cant use transitions to one history junction.	We can connect the number of transitions to one junction.
No need to use transitions to connect the states to one another.	By using transition to a junction, we can connect the states.

5. **What is Simulink?**

   Simulink® is a software package for Modelling, Simulating and analyzing dynamic systems. It supports linear and nonlinear systems, Model in continuous time, sampled time, or a hybrid of the two.

6. **Limitations of rate limiter:**

   You cannot use a Rate Limiter block inside a Triggered Subsystem. Use the Rate Limiter Dynamic block instead.

Stateflow	Flowchart
Stateflow is sequential logic.	Flowchart is a combinatorial logic.
In case of stateflow default transition executes only single time.	In flowchart the execution of the default transition executes every single time.
Stateflow contain the states.	do not contains states.
Stateflow considers current as well as previous result.	Flowchart does not consider previous result.
It contains entry, during, exit	It contains transitions & junctions.

7. **Why we use variable solver??**

Step size varies from step to step, depending on model dynamics.
A variable-step solver:
1.Reduces step size when model states change rapidly, to maintain accuracy.
2.Increases step size when model states change slowly, to avoid unnecessary steps.

8. **Why we choose fixed discrete solver?**

Because our microcontroller works at a fixed freq. and with time it cannot change. If our sample time is fixed then microcontroller will work properly. The step size will be constant throughout the simulation. The accuracy and the length of time of the resulting simulation depends on the size of the steps taken by the simulation: the smaller the step size, the more accurate the results are but the longer the simulation takes. The fixed-step discrete solver computes the time of the next simulation step by adding a fixed step size to the current time.

9. **What is algebraic loop error?**

An algebraic loop error occur when the output of a driven block is directly connected to the input of the same block. This error can be resolve by connecting the unit delay block between the input and output signal. Unit delay is a memory block that delay and hold it's input signal.

10. **How to solve algebraic loop error?**

To solve algebraic loop error we can use unit delay block & Memory block.

11. **How we choose between fixed and variable solver?**

Step size varies from step to step, depending on model dynamics.
A variable-step solver:
1.Reduces step size when model states change rapidly, to maintain accuracy.
2.Increases step size when model states change slowly, to avoid unnecessary steps.
Fixed-step -: Step size remains constant throughout the simulation.

12. **Why we create a subsystem??**

A subsys is a set of blocks that you can replace with a single subsys block. As your model increase in size and complexity, you can simplify it by grouping blocks into subsystem.

13. **What is simscape?**

14. **What is execution time.**
To measure the time required to run a function use the timeit function.

15. **What is data types?**

Data types supported by Matlab

- Name Description
- -------- ----------
- double - double precision, floating point numbers (8 bytes per element)
- uint8 -unsigned 8-bit integers in the range [0-255] (1 byte per element)
- uint16 - unsigned 16-bit integers in the range [0,65535]
- uint32 - unsigned 32-bit integers in the range [0,4294967295]
- int8 - singed 8-bit integers in the range [-128,127]
- int16 - singed 16-bit integers in the range [-32768,32767]
- int32 - singed 32-bit integers in the range [-2147483648,2147483647]
- single- single precision floating-point numbers (4 bytes per element)
- char - characters (2 bytes per element using Unicode rep.)
- logical - values are 0 or 1 (1 byte per element)
- All numeric computations (first 8 entries in the above table) in MATLAB are done using double quantities.
- uint8 is the frequently used 8-bit image.

Saturation	Dynamic saturation
It has 1 input.	It has 3 inputs.
We can set lower & upper limit inside the saturation box.	We can set lower & upper outside the dynamic saturation box.
We cant connect any logic directly to the saturation block. Cant use any other's logic as a input of saturation logic.	We can connect any logic to dynamic saturation. It means we can use any output as a input of dynamic saturation.

Virtual	Non-Virtual
Non-Atomic subs are virtual.	Atomic subs are non-virtual.
Code-generatn is not possible in virtual.	Code-generatn is possible in non-virtual.
Function packaging in virtual cant be control.	Function packaging in can be control in non virtual.

16. **Limitations Of VS**
    - Variant subsys, only one subsys can be active at a time.
    - We can not activate subsys. by one another.
    - To activate the inactive subsys, we have to give the condition in variant manager.
    - Also we have to ensure that the condition does not match the model name.

17. **Where we get Errors in VS?**
    - ''In log box'' we get the errors of VS.

18. **What is matlab? Why we use matlab??**

    MATLAB is a high−performance language for technical computing. Matlab is a programming platform which is designed specifically for engineers and scientist to analyse and design the products and systems that transforms our world. It integrates computation, visualization, and programming in an easy−to−use environment where problems and solutions are expressed in familiar mathematical notation.

    Typical uses include:
    ▪ Math and computation
    ▪ Algorithm development
    ▪ Modelling, simulation, and prototyping
    ▪ Data analysis

19. **What are matlab sys parts?**

    1.matlab lang. 2. Matlab working environment 3. Handle graphics 4. Matlab mathematical function library 5.Matlab application program interface.

20. **Why we use matlab?**

    Matlab is a wide tool. It is having application in various field like in medical science it is use monitor ECG or other healthcare data, and within less time it gets the report or results. In defence system it is use to monitor radar sys. **In our application we are using in automobile industry.** Day by day modifications are going on. Suppose in car, roof top was not there, automatic locking sys was not there but now roof top is there, with single button we can lock the whole car, open or close the window with one button. So these are the changes has been done in car with the help of matlab.
    And there is no need to have basic knowledge of C or java or python. The process of development is not complicated compare to C code or java. And that's why we are using basic blocks and we built our system easily.
    Auto code generation-:we don't have to write code manually we just need to focus on logic development, due to auto code generation it generate standard code so the coding which varies person to person that is avoided.
    Graphical design-Because visualization of logic is better than reading of thousand line of code we can easily understand what logic is doing.

21. **Why we pass the Maab guidelines?**

In maab guidelines, we have to specify the naming of model and usage of every block, so that client or some other new person from our organization will get the exact idea what we are doing in our model or project or what we were doing in the project. **With the help of maab guidelines project readability of model will be easy, easily accessible to other person or client.**

22. **What is MCDC/model coverage?**

Coverage means we do coverage analysis for the purpose of memory utilization and optimization. Whenever we do any change in any of the input then it affects on the output outcome. In this fully covered block shows green that means our conditions or decisions are satisfied at least once. If it shows red blocks then we are not meet with each condition or decision with our test input.

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

Decision Coverage (DC): Decision coverage analyzes 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.

Condition Coverage (CC) Condition coverage analyzes 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.

Modified Condition/Decision Coverage (MCDC) 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.
We are not able to get 100% mcdc we are getting 80% of mcdc in our industry. Because if we are having some dead logic we cant remove or replace it. If we do remove then if affects on required output.

23. **What is Mil?**

Model In Loop : In this we check model functionality. We can do mil testing by using signal builder block from to workspace block. But in our industry we prefer signal builder block to do a mil testing.

24. **What is Model in Loop:** When we develop the model and verify if the controller can control the plant as per the requirement.

25. **What is Sil?**

Software In Loop- With Embedded Coder®, you can run software-in-the-loop (SIL) of your model. A SIL simulation compiles and runs the generated code on your development computer. After successful mil testing we start for Sil. If our expected result match with sil result. Then our sil testing becomes successful. **S- Function block provides a computer lang. description to matlab in the form of block. It converts generated code understandable to matlab during sil testing.**

26. **Why we do Mil & Sil?**

We do Mil testing to check the model functionality and we do Sil to check the code functionality. i.e we do compare results i.e expected op with mil& sil and ensures that we designed the proper model and if we have generated proper code.

27. **Why we do Sil if we already did Mil?**

We do Mil testing to check the model functionality and we do Sil to check the code functionality. Mil is procedure where we compare the simulation op with expected op but in real time we are using the generated code to deploy in microcontroller or hardware system to check if it is working fine or not. To deploy the code in h/w sys we need a generated code and to generate the code we need a compiler so with the help of compiler we do sil testing and generate the code and deploy the code. So we are Mil testing in double precision & and sil we are using single precision input. So output may come different.

28. **Errors in Mil/Sil:**

Mil: a] Import file error (Index in position 2 exceeds array bounds,must not exceed 2)
b] Excel sheet blank: file xlsx format does not comply with signal builder reqd format.
c] undefined variable '' ans or out''
d] data overflow error e]data mismatch error

Sil: a] small quantization errors.
Model gets compiler errors.
b] precision loss- data type of model does not have enough precision to represent that parameter value. We need to set fixed dt to meet required precision value.
c] error in s-function-sys does not exist (before generating the s-function if we take s-function block).
d] If we join wrong inputs or interchange the inputs to s-function then we get mil & sil result in new excel file.
e] If we don't give right location/path, s-function or mil sil results appear at wrong location or with new excel file.
f] If we don't connect from and goto then we get ''matching does not found)
g] while generating MCDC if we open another location/path then we get error as ''s-funtion does not found''.

Condition	Decision	Execution
Relational operator	Relay	s-function
Logical operator	Switch	To workspace
Enabled , triggered subsystem	Multiport switch	Constant
Truth table	Dynamic saturation	product
Matlab function	Dead zone	addition
C/C++ Sfunction	Wrap to zero	Unit delay
Compare to zero, compare to constant	chart	Signal builder

29. **For variable which system target file we use?**

Rapid simulation target link (rsim.ert)

30. **Why we use ert instead of grt?**

For fixed step solver we use ert, it has small code size, it takes less memory compare to grt, it is compact, and better for speed and memory.

31. **For Sil which compiler we use? = Mingw64**

32. **MAAB =** Mathwork Automotive Advisory Board

33. **MISRA =** Motor Industry Software Reliability Association

34. **Tool for Maab & Misra –** Model Advisor
35. **Ert- Embedded Codder , Grt- Generic real time**
36. **Can we do Sil without Mil ?-** NO
37. **In ISO how many Asil Levels – Asil A ,Asil B, Asil C, Asil D**

Continuous solver	Discrete Solver
continuous solvers use numerical integration to compute the continuous states that the blocks define.	Discrete solvers exist primarily to solve purely discrete models.
Continuous solvers rely on the individual blocks to compute the values of the model's discrete states at each time step.	responsible for computing the values of those states at each time step.

38. **SLDV Simulink Design Verification.**

SLDV is the static verification tool which will help us to identify hidden errors, errors like
Integer flow errors
Dead logic
Divided by zero
Algebraic loop error
Also it generates auto testcase to achieve 100% coverage.
Maximum time= 300ms
Maximum test cases = 1000
SLDV also generates harness model.
Harness Model: It is the dummy model, in which we can make changes in that harness model and run the model without modifying the main model.

39. **What is default transition in stateflow?**
When there is number of states then default transition is the path to activate the first state.

40. **What is transition ?**
Transition is a line with an arrowhead that typically connects the 2 states.

41. **Why testing is important?**
The testing is important since it discovers the defects/bugs before delivery to the customer & which guarantees the quality of the software. It makes software more reliable and easy to use. Ensures the reliability and high performance software operations.

42. **Why we use stateflow?**
Stateflow used to respond instantaneous changes. In stateflow we can check or execute conditional logic easily.

43. **Procedure which you follow?**
Model Design- MAAB- code gen- sldv-testharness-mil-sil- Coverage report

Mil	Sil
Model In Loop	Software In loop
To check the functionality of Model.	To check the functionality of Code.
Mil is double precision in ip.	Sil is single precision in ip.
We perform mil testing before sil testing.	After mil testing we perform the sil testing.

44. **Why we use stateflow instead of Simulink??**

If you have to implement logic in simulink your model becomes usually difficult to be analysed after coding, but theoretically you can do anything in simulink. We can control event in stateflow. Debugging is easy in Simulink but readability and code generation are easy in stateflow.

45. **What is loop??**

Loops are used when task to be repeated no of times.

Black Box Testing	White Box Testing
It is used to test software without knowing the internal structure of code or program.	Internal structure of code or program is being known to a tester while testing the software.
This testing is carried out by tester.	Generally this type of testing is carried out by software developers.
Implementation knowledge is not required to carry out black box testing.	Implementation knowledge is required to carry out white box testing.
Programming knowledge is not required to carry out black box testing.	Programming knowledge is required to carry out white box testing.
BBT means functional and external testing.	WBT means structural and interior testing.

46. **What is Temporal Logic??**

In this state actions and transition actions are performed. There are two types 1. absolute based and 2.event based

SLDV	MCDC
Simulink design verifier.	Modified condition decision condition
We can generate auto testcases in sldv.	We can not generate auto testcases in mcdc.
The maximum time is 300ms.	There is no max time.
We can get dummy block in sldv.	We don’t get any dummy block.
Its a static analysis.	It’s a dynamic analysis.

47. **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 behavior of the model.

Four Sections are as follows:

Design Data

Configuration

Embedded coder dict

Other data

48. **V-Model**

Business Requirement Analysis

In this we gather requirements and expectations of customer.

This is the first phase in the development cycle where the product requirements are understood from the customer's perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirement. This is a very important activity and needs to be managed well, as most of the customers are not sure about what exactly they need. The **acceptance test design planning** is done at this stage as business requirements can be used as an input for acceptance testing.

System Design

It contains complete hardware and communication setup for developing product project.

Once you have the clear and detailed product requirements, it is time to design the complete system. The system design will have the understanding and detailing the complete hardware and communication setup for the product under development. The system test plan is developed based on the system design. Doing this at an earlier stage leaves more time for the actual test execution later.

Architectural Design

In this design is broken down further into modules taking up different functionalities.

Architectural specifications are understood and designed in this phase. Usually more than one technical approach is proposed and based on the technical and financial feasibility the final decision is taken. The system design is broken down further into modules taking up different functionality. This is also referred to as **High Level Design (HLD)**.

The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood and defined in this stage. With this information, integration tests can be designed and documented during this stage.

Module Design

In this phase the system breaks down into small modules.

In this phase, the detailed design of modules is specified, also known as **Low Level Design (LLD)**. It is important that the design is compatible with the other modules in the system architecture and the other external systems. The unit tests are an essential part of any development process and helps eliminate the maximum faults and errors at a very early stage. These unit tests can be designed at this stage based on the internal module designs.

Coding Phase

The actual coding of the system modules designed in the design phase is taken up in the Coding phase. The best suitable programming language is decided based on the system and architectural requirements.

The coding is performed based on the coding guidelines and standards. The code goes through numerous code reviews and is optimized for best performance before the final build is checked into the repository.

Validation Phases

The different Validation Phases in a V-Model are explained in detail below.

Unit Testing

Unit test plans are developed during model design phase. These unit test plans are executed to eliminate bugs at code or unit level.

Unit tests designed in the module design phase are executed on the code during this validation phase. Unit testing is the testing at code level and helps eliminate bugs at an early stage, though all defects cannot be uncovered by unit testing.

Integration Testing

After completion of unit testing int testing is performed. Integration tests are performed on the architecture design phase. Integration tests are performed to test the coexistence and communication of the internal modules within the system.

System Testing

System testing is directly associated with the system design phase. It testes functional and non functional requirements of the developed application. Complete application with its functionality, dependency and communication will test. System tests check the entire system functionality and the communication of the system under development with external systems. Most of the software and hardware compatibility issues can be uncovered during this system test execution.

Acceptance Testing

Acceptance testing is associated with the business requirement analysis phase and involves testing the product in user environment. Acceptance tests uncover the compatibility issues with the other systems available in the user environment. It also discovers the non-functional issues such as load and performance defects in the actual user environment.

The following pointers are some of the most suitable scenarios to use the V-Model application.

- Requirements are well defined, clearly documented and fixed.
- Product definition is stable.
- Technology is not dynamic and is well understood by the project team.
- There are no ambiguous or undefined requirements.
- The project is short.

49. 1.**Derivative**-The Derivative block approximates the derivative of the input signal u with respect to the simulation time t.
    2) **Integrator-**The Integrator block outputs the value of the integral of its input signal with respect to time.
    3)**Transport Delay**: The Transport Delay block delays the input by a specified amount of time.

4) **PID Controller**-The Discrete PID Controller block implements a PID controller (PID, PI, PD, P only, or I only).

5)**Zero pole**- The Zero-Pole block models a system that you define with the zeros, poles, and gain of a Laplace-domain transfer function.

6**) Transfer fcn-** The Transfer Fcn block models a linear system by a transfer function of the Laplace-domain variable s.

50. Check for usage of Merge block.

Condition	Recommended Action
One or more blocks inserted in between a Merge and a Conditional Subsystem block.	Make direct connections from Conditional Subsystem blocks to Merge blocks.

51. Check usage of Switch blocks

Condition	Recommended Action
The Switch block control input (second input) is not a Boolean value.	Change the data type of the control input to Boolean.

Condition	Recommended Action
The Switch block is not configured to pass the first input when the control input is nonzero.	Set the block parameter **Criteria for passing first input** to u2 ~=0.

52. Check usage of Relational Operator blocks

Condition	Recommended Action
Relational Operator blocks have a Constant block on the first, upper input.	Move the Constant block to the second, lower input.

53. Check usage of the Saturation blocks

Condition	Recommended Action
The input and output data types are different.	Make sure that the **Output data type** is set to **Inherit: Same as input** and **Inherit: Same as second input** for Saturation and Saturation Dynamic blocks respectively.
The upper limit and lower limit values of the blocks are not set to adhered values.	• Set the **upper limit** of the output data type to less than the maximum value. • Set the **lower limit** of the output data type to less than the minimum value.

54. Check usage of Memory and Unit Delay blocks

Condition	Recommended Action
Memory blocks have sample time that is not continuous.	Use Unit Delay block instead of Memory block.
Unit Delay blocks have non discrete sample time.	Use Memory block instead of Unit Delay block.

55. **Why we use ISO 26262 guidelines? - International Standard Organization**

Its is an international functional safety standard titled *Road vehicles — Functional safety*. ASIL A represents the lowest degree and ASIL D represents the highest degree of automotive hazard. Systems like airbags, anti-lock brakes, and power steering require an ASIL-D grade—the highest rigor applied to safety assurance—because the risks associated with their failure are the highest.

Asil- Automotive Safety Integrate Level.

Asil A- rare light, vision Adas

Asil B- Headlight, brake lights, rare view cameras

Asil C- engine management, active suspension, radar cruise control

Asil D- air bags, electric power steering, antilock braking

56. **ISO 26262 Guidelines**

Simulink:

1. Check usage of Merge blocks
2. Check usage of Relational Operator blocks
3. Check usage of Logical Operator blocks
4. Check usage of bit operation blocks
5. Check usage of variant blocks
6. Check usage of lookup table blocks
7. Check usage of Assignment blocks
8. Check usage of Gain blocks

Stateflow:

1. Check Stateflow debugging options
2. Check naming of ports in Stateflow charts
3. Check scoping of Stateflow data objects
4. Check usage of bitwise operations in Stateflow charts
5. Check assignment operations in Stateflow charts

Matlab:

1. Check MATLAB Function metrics
2. Check MATLAB Code Analyzer messages
3. Check switch statements in MATLAB Function blocks
4. Check usage of relational operators in MATLAB Function blocks
5. Check usage of equality operators in MATLAB Function blocks

Configuration:

1. Check safety-related model referencing settings
2. Check safety-related solver settings for solver options
3. Check safety-related diagnostic settings for solvers
4. Check safety-related diagnostic settings for sample time
5. Check safety-related code generation symbols settings
6. Check safety-related diagnostic settings for Merge blocks
7. Check safety-related diagnostic settings for bus connectivity
8. Check safety-related diagnostic settings for signal connectivity
9. Check safety-related diagnostic settings for Stateflow
10. Check safety-related diagnostic settings for signal data

Bug Reports:

1. Display bug reports for Embedded Coder
2. Display bug reports for IEC Certification Kit
3. Display bug reports for Polyspace Bug Finder
4. Display bug reports for Simulink Check
5. Display bug reports for Simulink Coverage
6. Display bug reports for Simulink Test
7. Display bug reports for Simulink Requirements

Code:

1. Check for blocks not recommended for MISRA C:2012
2. Check configuration parameters for MISRA C:2012

Display model metrics and complexity report

Display configuration management data

Check for unconnected objects

57. **Modeling Standards for MISRA C:2012**

1. Check for blocks not recommended for MISRA C:2012
2. Check configuration parameters for MISRA C:2012
3. Check for unsupported block names
4. Check usage of Assignment blocks
5. Check for recursive function calls
6. Check integer word lengths
7. Check for bitwise operations on signed integers

58. **Why we pass the MISRA Guidelines?**

The MISRA C and MISRA C++ standards are a set of coding guidelines for the C and C++ programing languages that promote safety, security, and reliability in embedded system software.

59. **Diff bet scalar, vector & matrix?**

Scalar a number like 3, -5, 0.368, A vector is a list of numbers (can be in row or column), A matrix is an array of numbers(one or more rows or column)

Scalar= 25 , Vector= [6 45 12], matrix= $\begin{matrix} 12 & 2 & 30 \\ 5 & 25 & 18 \end{matrix}$

60. **Why we do unit testing if we are doing integration testing?**

Bcoz its difficult to debug the model after in integration tes. So individual unit testing we do bcoz model is complex. Unit test is earliest phase, its easy to find bug at ut. If we merge all then debug then it will be complex procedure. And also it will be time consuming process.

61. **MAAB GUIDELINES**

Naming Conventions

1. Check file names
2. Check folder names
3. Check subsystem names
4. Check port block names
5. Check character usage in signal labels
6. Check character usage in block names
7. Check Simulink bus signal names

Model Architecture

1. Check for mixing basic blocks and subsystems
2. Check unused ports in Variant Subsystems
3. Check use of default variants
4. Check use of single variable variant conditionals

Model Configuration Options

1. Check Implement logic signals as Boolean data (vs. double)
2. Check model diagnostic parameters

Simulink

1. Check for Simulink diagrams using nonstandard display attributes
2. Check font formatting
3. Check positioning and configuration of ports
4. Check visibility of block port names
5. Check display for port blocks
6. Check whether block names appear below blocks

7. Check the display attributes of block names
8. Check for nondefault block attributes
9. Check for matching port and signal names
10. Check Trigger and Enable block names
11. Check **signal line labels**
12. Check for propagated signal labels
13. Check for prohibited sink blocks
14. Check for indexing in blocks
15. **Check usage of Switch blocks**
16. **Check usage of Relational Operator blocks**
17. Check usage of buses and Mux blocks
18. Check usage of non-compliant blocks
19. **Check usage of Goto and From blocks between Subsystems**
20. **Overlapping of signal**
21. **Executional order manage**
22. **Sub and basic blocks not be in same level**
23. **Check usage of Goto and From blocks should be in one level**

Stateflow

1. **Check for pointers in Stateflow charts**
2. Check use of Simulink in Stateflow charts
3. Check Stateflow transition appearance
4. Check number of Stateflow states per container.
5. **Check for bitwise operations in Stateflow charts**
6. Check reuse of Variables within a Stateflow scope
7. Check for comparison operations in Stateflow charts
8. Check for event broadcasts in Stateflow charts
9. **Uniqueness of state names**
10. **Length of stateflow data names**
11. **Comments in state names**
12. Scope of data in parallel actions
13. Conditions should be horizontal , vertical- actions because it will create the loop. It will not understand no of iterations
14. Always there should be for action ; , conditions should be in []

MATLAB Functions

1. Check MATLAB Function metrics
2. Check MATLAB code for global variables
3. Check usage of restricted variable names
4. Check the number of function calls in MATLAB Function blocks

62. **What is state machine??**

State machine is any device that stores the status of something at a given time.

63. **V methodology…y??**

There are two phases validation and verification, and this process work simultaneously, which is the time consuming process.

64. **What is bug finder and code prover**

bug finder –finds the bug in the report

code prover- it proves the accuracy of the code.

65. **What is baseline?**

Within the deadline we complete our task to supplier

66. **Model Hierarchy Pane**

The Model Hierarchy pane displays a tree-structured view of the Simulink model hierarchy.

67. **What is pointers**

A pointer is a variable that stores address of another variable.

68. **Enumeration**

Enumeration is user defined in data type C. It is mainly used to assign names to the integral constant, the names make a program easy to read and maintain.

69. **Super transition**

A super transition is a transition between different levels in a chart.

70. **Simulink Function**

A Simulink® function is a graphical object that you fill with Simulink blocks and call in the actions of states and transitions.

71. **Graphical function**

A graphical function in a Stateflow® chart is a graphical element that helps you reuse control-flow logic and iterative loops. You create graphical functions with flow charts that use connective junctions and transitions. You can call a graphical function in the actions of states and transitions. With graphical functions,

72. **What is lookup table??**

Lookup table is an array of data that maps input values to output values. simply we can say

We can estimate the model with the help of lookup model.

If we provide input values to the sys.then sys will provide op values.

Using this data we convert one signal to another signal.

73. **Interpolation and Extrapolation**

An **interpolation** is a process of forestimating the values that lie between the known data points.

An **extrapolation** is a process of forestimating the values that lie beyond the known data points.

74. **CAN**

It is a serial half duplex asynchronous communication protocol. Messaged based protocol.
It's a broadcast type of bus. Which is designed for microcontroller and devices to communicate with each other without need of the host.
Here is no doubt if all communicating nodes are connected one to one. The speed and communication will be high but network complexity and cost of wires and connectors will also be high. We can connect 30-40 nodes to one node. This is a linear bus structure.
Can introduced as a centralised solution which required CAN high and CAN low.
Can **versions** are

1) Can 2.0 (zero)A- standard 11 bit
2) Can 2.0 B- extended 29 bit

Types of Fields

1) Arbitration field
2) Control field
3) Data field
4) CRC
5) acknowledgement

Can Frames:

1. Data frame
2. Remote frame
3. Error frame
4. Overload frame

Errors of CAN

Bit Error
CRC error
Ack error
Bit stuffing error

75. DB **Condition action transition action**

Condition action	**Transition action**
1. Condition action execute every time when the condition evaluate true.	1. Transition action executes when transition occurs valid.
2. This is only depends on the condition.	2. This is only depends on the transition.
3. Condition action specifies before transition action.	3. Transition action specifies before condition action.

76. **Conditional executed sub??**

A nonvirtual sub that allows you to control its execution with an external signal.

77. **What is Solver?**

Simulink® provides a set of programs called solvers. A solver applies a numerical method to solve the set of ordinary differential equations that represent the model. Through this computation, it determines the time of the next simulation step.

78. **What is Sample time?**

Sample time is a time taken by block to updates its internal state. Sample time should be greater than 0 and less than simulation time. Sample times can be port based or block based. For block−based sample times, all of the inputs and outputs of the block run at the same rate.For port−based sample times, the input and output ports can run at different rates

79. **Define step size?**

The size of the interval is known as the step size. You can specify the step size or let the solver choose the step size. Generally, decreasing the step size increases the accuracy of the results while increasing the time required to simulate the system

81. **Enable control signal has a positive value.** In Enabled subsystem there are 2 condition -:

a) Held (Hold) -: States maintain their most recent values. (it holds the state until next triggered value occur) b) Reset -: States revert to their initial conditions if the subsystem is disabled for at least one time step.

82. **What Is Variant Subsystem?**

Ans-: Variant subsystems let you provide multiple implementations for a subsystem where only one implementation is active during simulation. You can programmatically swap out the active implementation and replace it with one of the other implementations without modifying the mode.

83. **Tools in matlab:**

1. Simulink
2. stateflow
3. Embedded coder
4. SLDV
5. Signal builder
6. Model adviser

84. **Static and Dynamic**

Static Analysis	**Dynamic Analysis**
In this we don't decide how much time it will required to run the whole model.	In this with respect to time model runs.
The testcases can generates automatically.	We write the testcases it means we manually generate the testcases.
Polyspace, sldv	Simulink, mil

85. Microcontroller and microprocessor

Microprocessor is a processor where the memory and I/p component are connected externally. A microcontroller is a controlling device wherein the memory and I/p op component are present internally.

Matlab is a assembly language. That's we convert it to ert.code.

Sil- we can do sil but we will not find errors in sil. Process after mil, standard procedure…

86. What's the garbage value??

If a variable is assigned but not allocated in some programming lang. such as C.

Black Box Testing	White Box Testing	Gray Box Testing
No knowledge of the internal structure	The internal structure is completely known	The Internal structure is partially known
Also known as closed box, functional testing, and data-driven testing	Also known as glass, clear box, structural testing, or code-based testing	Also known as translucent testing
Players are testers, developers, and the end-users	Players are testers and developers only	Done by testers, developers, and the end-users
Not suitable for algorithm testing	Suitable for algorithm testing	Not suitable for algorithm testing
Tester has nothing to do with the internal working of the system	Tester has complete knowledge of the internal working of the system	Tester has partial knowledge of the internal working of the system
Hidden errors cannot be found easily	Hidden errors can be found very easily as internal working is well known to the tester	Not easily found but can be found at the user level
Least time-consuming process	Very time consuming	Gray box testing takes less time than white box testing
Less exhaustive than the white box and gray box testing	Most exhaustive process	Partially exhaustive process

Array	Pointer
An array is a defined as a collection of similar datatypes.	Pointer is a variable which stores address of another variable.
The nature of arrays is static.	The nature is pointer is dynamic.
The allocation of array is done at compile time.	The allocation of pointer is done at run time.
Arrays cannot be resized according to users requirements.	Pointers can be resized at any point of time.

87. BCM

Sensors used:

1. Speed Sensor- It measures the speed of the wheels.
2. Parking sensor- uItrasonic sensor-recognises any obstacle present in the front or rear of the vehicle.
3. Rain sensor- It detects rain and sends a signal to ECU to activate the wipers.
4. For curve road- GPS, front facing camera, IR sensor- Infrared
5. Distance, map-radar system, lidar, ultrasonic
6. Weather- moisture sensor
7. Brake light- DBL- Dynamic brake light
8. Proximity sensor- reverse light (when the vehicle gets too close to an object.)
9. Magnet sensor—door light

88**. Mil Testing**

Model In Loop : In this we check model functionality.
For Mil testing we are writing possible testcases in the excel sheet as per the req. then we are taking signal builder block, and importing that excel sheet of testcases. Connecting ip op to sub with signal builder block. Setting sim time as per the testcases written in excl sheet. That time we set. Then Run. Checking exp with act op. if its matched with exp and act then its successful.

For Sil we are using S function block. 1st generating s- function, after creating s function we are getting 3 files.(.c .h .mex) after that we are taking that mex file to s block.. after that signal builder and importing test cases…and run and the check exp with act op…if its matched then its successful

89**.**

For Loop- For loop is a control flow statement for specifying iteration.

While loop- It is entry controlled loop. In while loop 1st checking the condition if condition is true then only body of loop will get executed.

Syntax

While condition

{

Statement

}

Do while- It is similar to while loop but except do while is executed at least one time.

Do

{

Statement

}

While

}