1. (20%) What is static verification? What are the advantages and disadvantages of static verification?? Please also list two possible applications of static verification and explain their purpose briefly.

(i) Static verification is a technique used to check the correctness of a design or implementation without actually executing it.

(ii) Advantages: Improved code quality, early defect detection

Disadvantage: May show false positives, may not be able to analyze all aspects of a design

(iii)

I. Formal verification: using mathematical methods to ensure that a design is free from certain types of errors or to verify that a design meets certain functional requirements.

II. Linting: running "linter" over source code to check for style violations and potential errors.

III. Static analysis: analyzing source code without actually executing them. It can be used to find defects, security vulnerabilities, or other issues in a design or implementation.

IV. Model checking: constructing a model of the design and using automated tools to explore all possible states and transitions of the model to ensure that it meets the desired properties.

3. (15%) What are the keys to reduce power consumption at system level?? Can you list two possible techniques to reduce dynamic power at system level with brief discussion about their benefits and limitations?

(i) Reduce dynamic power or leakage power

(ii) Dynamic power = $C * Vdd^2 * f$

I. Clock gating – turn clocks off whey they are not required to reduce power consumption / increase the complexity of control logic

II. Multiple power domains – lower Vdd values for non-critical blocks / level shifters may increase delay, may increase the complexity of timing analysis

III. Dynamic voltage and frequency scaling – scale the operating voltage or frequency for different tasks / determining which voltage and clock values to support is complicated, increase verification complexity

4. (15%) Compared to original Verilog language, what are the major extensions of Verilog-AMS? How to improve the accuracy and efficiency of the behavioral models based on Verilog-AMS??

(i) Verilog-AMS adds support for analog and mixed-signal modeling, including continuous and discrete time modeling, linear and nonlinear equations, etc.

(ii) Use appropriate value for each parameter for accurate simulation results and concise mathematical equations of the behavior for faster simulation time.

5. (15%) What are the difficulties of analog design automation? What are the benefits and limitations of equation-based optimization approach??

(i) Accurately modeling and simulating an analog circuit is challenging, since analog circuits are sensitive to variations in manufacturing processes, supply voltages, and temperature, which can make it difficult to predict their behavior accurately.

(ii) Numerical solvers can be applied to find a feasible solution with minimum cost (designers' knowledge and experience are not required, compared to knowledge-base optimization) / simplified equations and parameter models limit the accuracy of the result, compared to real simulation results.

6. (20%) Why does the difference between prediction and simulation often exist at circuit design stage? What are the issues and possible solutions to overcome the performance gap in analog design automation flow??

(i) Prediction models are typically based on assumptions and approximations that may not perfectly capture the behavior of real circuits.

(ii) Develop more accurate and sophisticated simulation models, such as putting parasitic effects into considerations, that better capture the behavior of real circuits.

1. (15%) What is power gating technique? What are the benefits of using this technique? Can we always gain power reduction by using power gating? Please briefly discuss the possible issues and limitations while using this technique.

(i) Power gating is a technique used to reduce the power consumption of a circuit by turning off the power supply to certain blocks or modules when they are not in use.

(ii) Reduce power consumption, decrease both leakage power and dynamic power

(iii) No, power gating is more effective at reducing static power consumption, and it may not be as effective at reducing dynamic power consumption. Besides, the performance may become worse due to IR drop effects and in-rush current during power-up could cause extra supply noise issues.

2. (15%) What are the advantages of using code coverage in simulation-based functional verification? If you got 100% statement coverage in the simulation, does it guarantee that the design is correct? Please explain the reasons of your answer.

(i) It is easy to use, has low complexity, points out areas of the code that have not been exercised by the test suites, and also minimize the use of simulation resources.

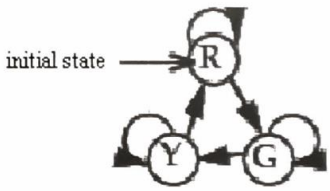
(ii) No, it only shows that all statements are executed and cannot guarantee 100% error-free design.

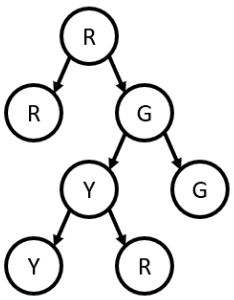
3. (15%) What are the advantages and disadvantages of formal verification? What are the keys to improve the efficiency of formal verification??

(i) Advantage: Fast and reliable, ensures consistency with specification for all possible inputs.
Disadvantage: Requires lots of memory to build BDDs, not suitable for large designs.

(ii) Simplifying the system being verified (such as using abstraction), utilizing simulation results to help improve formal verification efficiency.

4. (10%) Do the CTL formulas below satisfy the STG shown right ? Please answer "TRUE" or "FALSE".
(a) **EG**(RED)
(b) **AF**(GREEN)
(c) **E**(RED **U** GREEN)
(d) **A**(RED **U** GREEN)
(e) **AG**(RED → **EX**(GREEN))





(a) T, exist a path that is all 'RED'.

(b) F, for all paths, you can find a 'GREEN' in the future.

(c) T, exist a path that is all 'RED' until 'GREEN'.

(d) F, for all paths, it is all 'RED' until 'GREEN'.

(e) T, for all paths, if there is a 'RED', there exist a path that next state is 'GREEN'.

5. (10%) Using the CTL formulas, how can we verify the design functionality? What are the issues of CTL-based verification?

(i) Computation tree logic (CTL) is a temporal logic that can be used to specify properties of systems that change

over time. We can use CTL to check if the design satisfied all specs (paths).

(ii) The size of the state space grows exponentially as the complexity of the system increases. This may consume lots of memory and take lots of time to verify the design.

6. (10%) Why do we need analog behavioral models for mixed-signal system verification? Compared to the original Verilog language, what are the major extensions of Verilog-AMS?

(i) Verification via HSPICE takes lots of time. Behavioral models simplified the mixed-signal system to make the verification process more efficient and effective.

(ii) Verilog-AMS adds support for analog and mixed-signal modeling, including continuous and discrete time modeling, linear and nonlinear equations, etc.

7. (10%) Given a 4-phase sin-wave generator, please show the analog section of the behavioral model for this circuit based on Verilog-A. Its output voltage swing is 1.0V with 0.4V DC shift, and the phase difference between consecutive outputs is 90°. The signal frequency is 250MHz with maximum slew rate ±80MV/sec.

(Correctness not guaranteed)

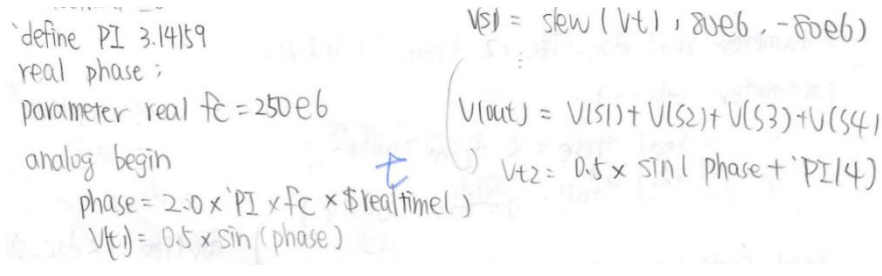
By ChatGPT

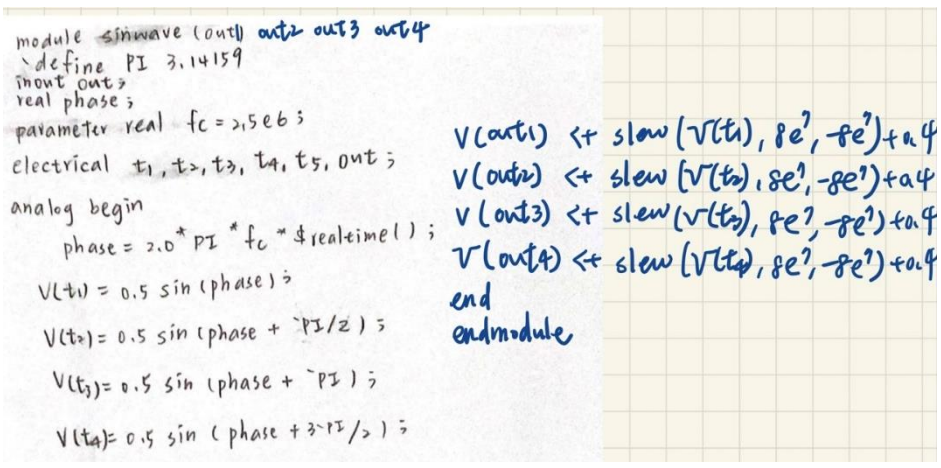
```
module sin_wave_4phase (input f, output [3:0] out);

parameter VPP = 1.0; // output voltage swing
parameter VDC = 0.4; // DC shift
parameter phase_diff = 90; // phase difference
parameter f = 250e6; // signal frequency
parameter slew_rate = 80e6; // maximum slew rate

real t;
real y;
real phase;

analog begin
   t = (2*$pi*f*time);
   for (phase = 0; phase <= 360; phase += phase_diff) begin
      y = VDC + (VPP/2)*sin(t + phase*(2*$pi/360));
      out[int(phase/90)] = y;
   end
end
endmodule
```





8. (15%) What are the benefits and limitations of simulation-based approach and equation-based approach for analog design automation? Why does the difference between synthesis results and post-layout simulation often exist?

(i) Simulation-based: can handle a wide range of design styles and circuit, and can provide more accurate result / may require a large number of simulations to achieve good accuracy, which takes lots of time.
Equation-based: can be more efficient than simulation-based approaches, as they can directly solve for the desired design parameters / may not be so accurate due to the approximations and assumptions of the modeling equations.
Pareto-front-based: They can provide a tradeoff between different design objectives, allowing designers to choose a design point that best meets their needs / may not be able to find the optimal solution, as the optimal solution may lie outside of the search space, may require a large number of design simulations in order to generate the Pareto front, which can be computationally intensive.

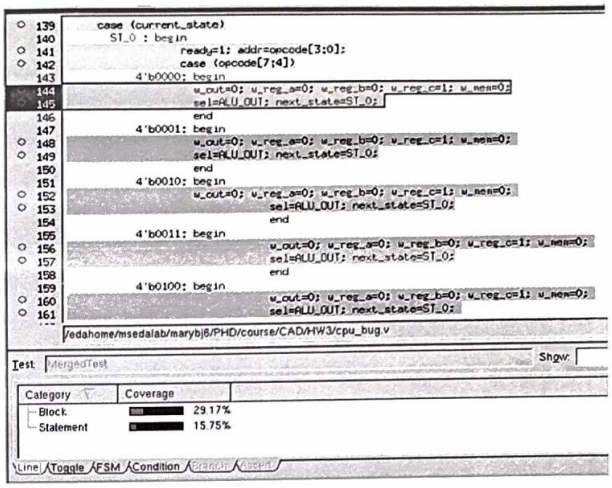
(ii) The synthesis process is not able take into account all of the parasitics and other effects that are present in the actual layout of the circuit.

1. (15%) What are the keys to reduce power consumption at system level?? If DVFS technique is applied in a design, does it reduce dynamic power or leakage power? Does it have any limitations? Please briefly explain your reasons.

(i) Dynamic power is reduced, as it can lower the voltage and frequency of the clock signal.

(ii) Determining which voltage and clock values to support is complicated, increase verification complexity, may not be suitable for systems with real-time constraints, as reducing the frequency of the clock signal may increase the execution time of certain tasks.

2. (15%) While you are running code coverage analysis, you got a coverage report as shown right. What are the meanings about the numbers in the bottom window and the red lines in the upper window? In your opinions, is the result good enough for functional verification? What can we do for the users who are not satisfied with the results?



(i) Some block and statements are not executed (reached).

(ii) No, 100% would be better. Users could run more test patterns to improve the result.

3. (15%) What are the benefits and drawbacks of using the simulation-based approach and formal-based approach for functional verification?? What are the keys to improve the efficiency of simulation-based verification??

(i) Simulation-based: better accuracy / may take a long time to generate accurate results
Formal-based: More efficient / not suitable to handle large design due to high complexity, not general solution

(ii) Use more efficient software or better hardware to simulate, parallelize the verification process

4. (15%) What is "equivalent checking" verification technique? What are the benefits and limitations of this technique?? Is it possible to have false alert in real cases with this technique??? Please explain your reasons.

(i) Equivalent checking checks for mismatches between two gate-level circuits or HDL and gate-level designs.

(ii) Solving SAT problem, gaining acceptance in practice / targets implementation errors, not design errors

(iii) No method is foolproof, and there is always a chance for human error or other factors that can lead to a false result.

5. (15%) What are the differences of using top-down and bottom-up approaches for analog behavioral modeling in mixed-signal system designs? How can they help designers in the design flow??

(i) Top-down approach: verify whole system roughly before implemented, suitable for new design
Bottom-up approach: more accurate, consider non-ideal effects, suitable for flattened transistor-level designs and IP-based designs.

(ii) The top-down approach helps develop a high-level understanding of the system behavior in early stage, while the bottom-up approach can be useful for verifying the accuracy of the model at a detailed level and identifying implementation-specific issues.

6. (10%) Given a 2-bit DAC circuit, please show the analog section of the behavioral model for this circuit based on Verilog-A. The maximum input voltage is 1.0V, and the signal frequency is 250MHz. The output signal has a 0.5ns rising time and a 0.4ns falling time, and its input-to-output delay time is 1ns.

(Correctness not guaranteed)

By ChatGPT

```
module dac_2bit (input IOV, input Ins, input [1:0] in, output out);

parameter IOV = 1.0; // maximum input voltage
parameter Ins = 1.0; // input-to-output delay
parameter f = 250e6; // signal frequency
parameter tr = 0.5e-9;
parameter tf = 0.4e-9;
analog begin
   VDAC0 = (in[0] == 1) ? IOV : 0;
   VDAC1 = (in[1] == 1) ? IOV : 0;
   Vout = VDAC0 + VDAC1;
   out = Vout;
end

endmodule
```

```
8. module ideal_dac (in,out);
   input [0:dac_size-1] in;
   output out;
   voltage in,out;
   parameter real dac_size =2 from (1:inf);
   parameter vth=2.5;
   +       real trise = 0 from [0:inf)
   +       real tfall = 0 from [0:inf)

   real code;
   integer pow2 [0:dac_size]
   analog begin
     @(initial_step)
       for (i=0; i<=dac_size; i=i+1) pow2[i]= pow(2,i)
```

```
code=0
for(i=0; i<dac_size; i=i+1)
code=code+(V(in[i]) >Vth
   ? pow2[i]:0;
V(out) <+ transition
(code/pow2 [dac_size
```

```
b.
analog begin

@(initial_step)
  pow2[0] = pow(2,0);
  pow2[1] = pow(2,1);

  code = 0;
  code = code + ( V(in[0]) > Vth) ? pow2[0] : 0;
  code = code + ( V(in[1]) > Vth) ? pow2[1] : 0;

  V(out) <+ transition ( code , In , 0.5n , 0.4n);
```

7. (15%) What are the benefits and limitations of simulation-based approach and equation-based approach for analog design automation? What are the possible bottlenecks for those circuit synthesis techniques to be applied on real cases?

(i) Simulation-based: accurate / time-consuming

Equation-based: faster, solvers can find feasible solution with minimum cost / not accurate

(ii) It takes lots of time to run many test patterns to get accurate result for simulation-based approach.

Simplified equations and approximations limited the accuracy of equation-based approach.

2-bit ADC Verilog-A code

(Correctness not guaranteed)

By ChatGPT

```
module adc_2bit (input IOV, input Ins, input in, output [1:0] out);
parameter IOV = 1.0; // maximum input voltage
parameter Ins = 1.0; // input-to-output delay
parameter f = 250e6; // signal frequency
parameter tr = 0.5e-9;
parameter tf = 0.4e-9;

analog begin
   if (in > IOV*3/4) begin
      out[1] = 1;
      out[0] = 1;
   end
   else if (in > IOV*1/4) begin
      out[1] = 0;
      out[0] = 1;
   end
   else if (in > IOV*-1/4) begin
      out[1] = 0;
      out[0] = 0;
   end
   else if (in > IOV*-3/4) begin
      out[1] = 1;
      out[0] = 0;
   end
   else begin
      out[1] = 1;
      out[0] = 1;
   end
end

endmodule
```