AutoChip_Report

1. Overview-BCD:

In this assignment, I used AutoChip to generate synthesizable Verilog for a 5-bit Binary-to-BCD converter. The design converts a binary input (0–31) into an 8-bit packed BCD output, where:

bcd_output[7:4] = tens digit

bcd_output[3:0] = ones digit

I selected this example as the combinational design task. The provided testbench compares the DUT against a golden reference implementation using:

tens = binary_input / 10;ones = binary_input % 10;

The AutoChip workflow iteratively generated RTL, compiled it using iverilog, simulated it, and used mismatch feedback to refine the design.

2. Initial Prompt and Configuration

Model Configuration

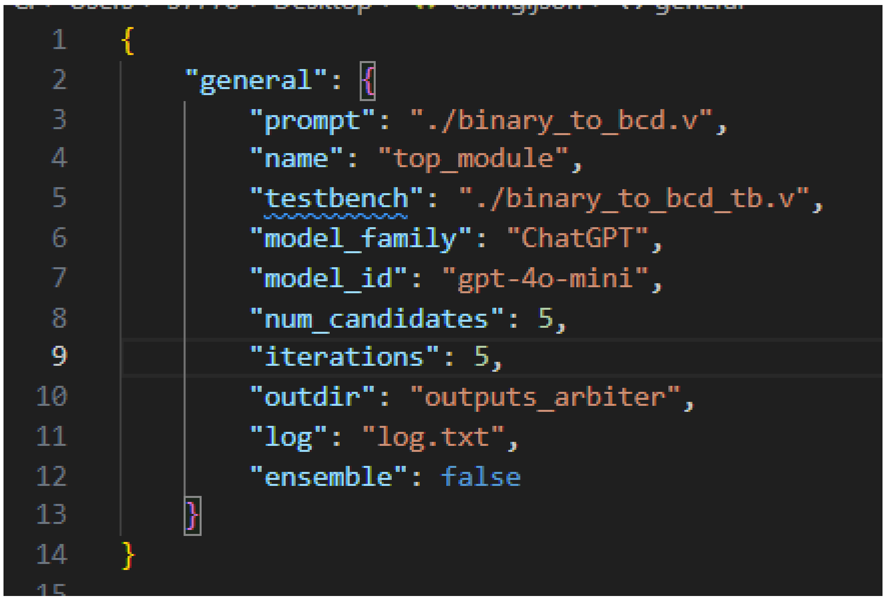
Model: gpt-4o-mini

Iterations: 5

Num candidates: 5

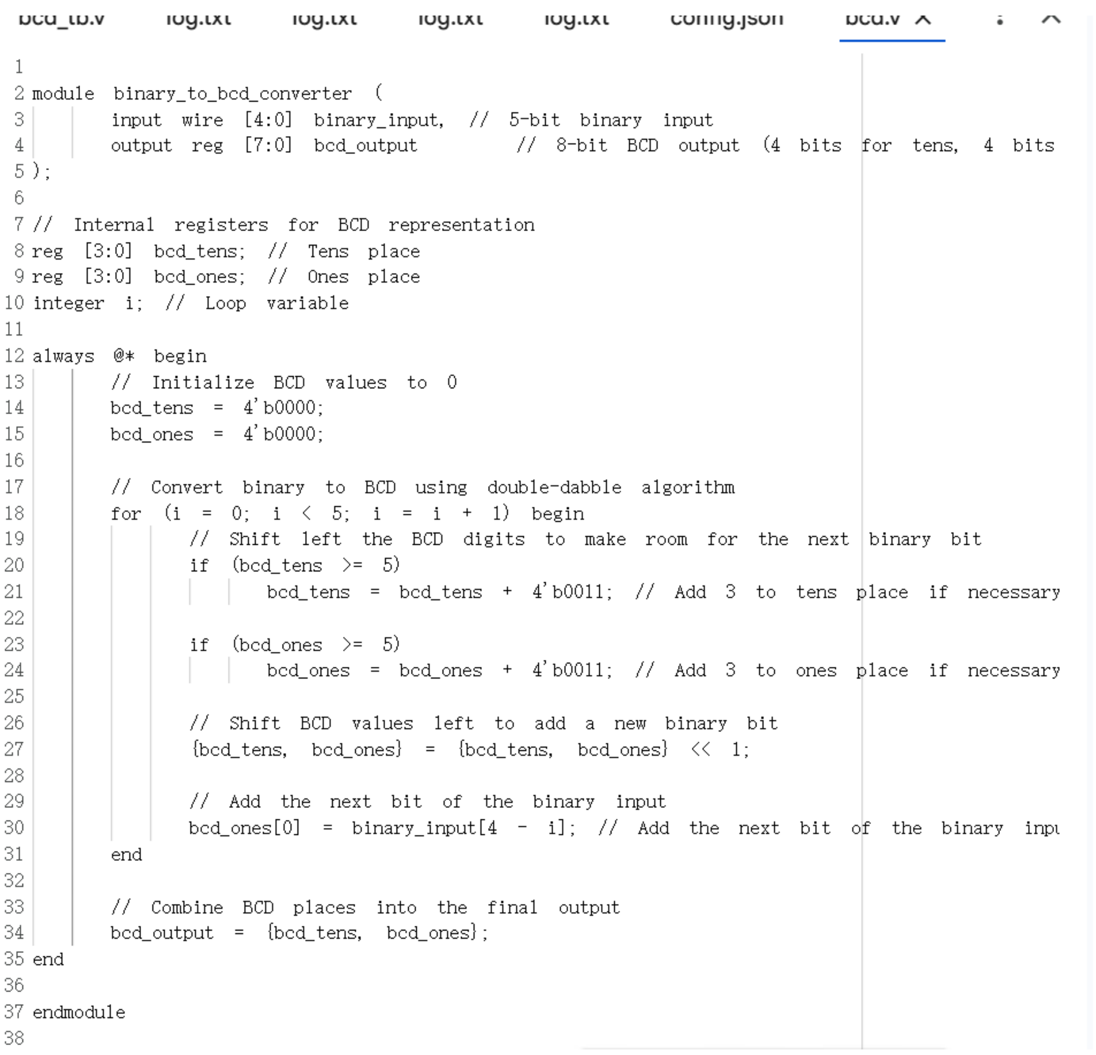
Ensemble: disabled

Toolchain: iverilog -g2012



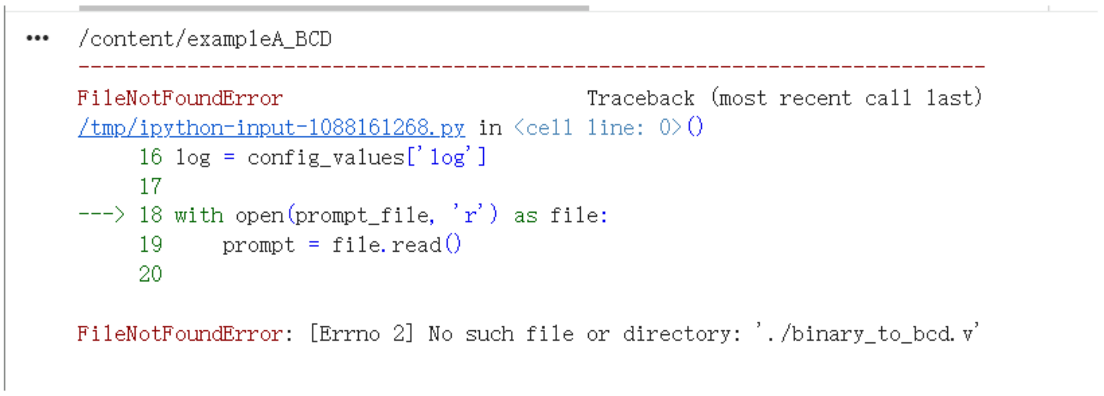
```
{
    "general": {
        "prompt": "./binary_to_bcd.v",
        "name": "top_module",
        "testbench": "./binary_to_bcd_tb.v",
        "model_family": "ChatGPT",
        "model_id": "gpt-4o-mini",
        "num_candidates": 5,
        "iterations": 5,
        "outdir": "outputs_arbiter",
        "log": "log.txt",
        "ensemble": false
    }
}
```

Initial Prompt



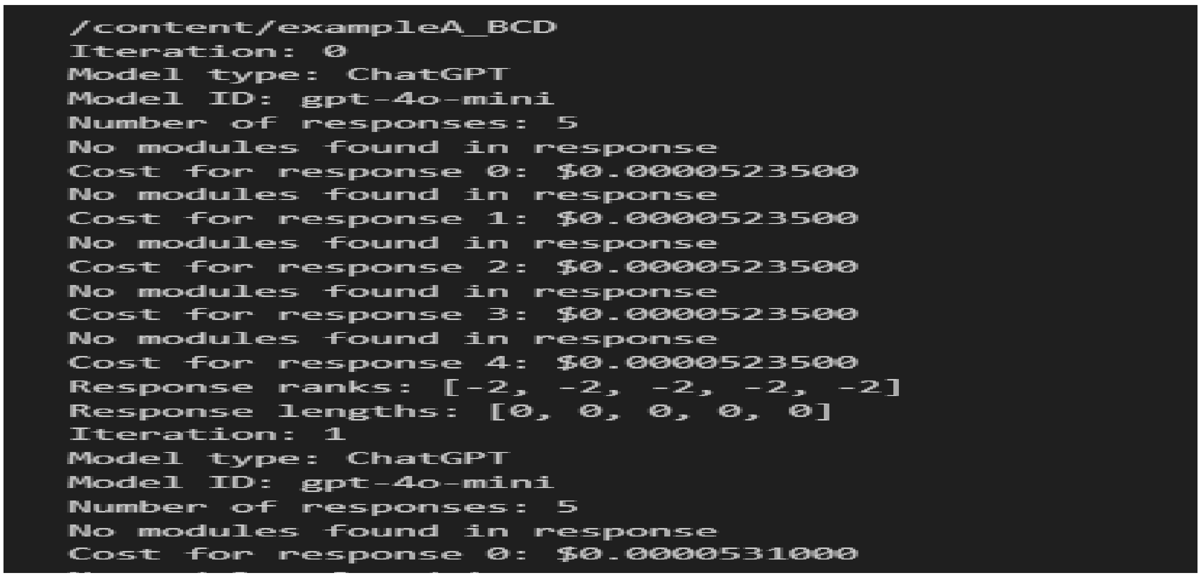
```

module binary_to_bcd_converter (
    input wire [4:0] binary_input,  // 5-bit binary input
    output reg [7:0] bcd_output        // 8-bit BCD output (4 bits for tens, 4 bits
);

// Internal registers for BCD representation
reg [3:0] bcd_tens; // Tens place
reg [3:0] bcd_ones; // Ones place
integer i; // Loop variable

always @* begin
    // Initialize BCD values to 0
    bcd_tens = 4'b0000;
    bcd_ones = 4'b0000;

    // Convert binary to BCD using double-dabble algorithm
    for (i = 0; i < 5; i = i + 1) begin
        // Shift left the BCD digits to make room for the next binary bit
        if (bcd_tens >= 5)
            bcd_tens = bcd_tens + 4'b0011; // Add 3 to tens place if necessary

        if (bcd_ones >= 5)
            bcd_ones = bcd_ones + 4'b0011; // Add 3 to ones place if necessary

        // Shift BCD values left to add a new binary bit
        {bcd_tens, bcd_ones} = {bcd_tens, bcd_ones} << 1;

        // Add the next bit of the binary input
        bcd_ones[0] = binary_input[4 - i]; // Add the next bit of the binary inpu
    end

    // Combine BCD places into the final output
    bcd_output = {bcd_tens, bcd_ones};
end

endmodule

```

Actually my first prompt include some logic embedded into the module and logic hint for the generation.



```
/content/exampleA_BCD
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
/tmp/ipython-input-1088161268.py in <cell line: 0>()
     16 log = config_values['log']
     17 
---> 18 with open(prompt_file, 'r') as file:
     19     prompt = file.read()
     20 

FileNotFoundError: [Errno 2] No such file or directory: './binary_to_bcd.v'
```

But the first time i didn’t pronounce the right file name. I mistake the binary_to_binary.v with binary_to_binary.sv so i change the file back again to .v to ensure there is no conflict and solve the problem.

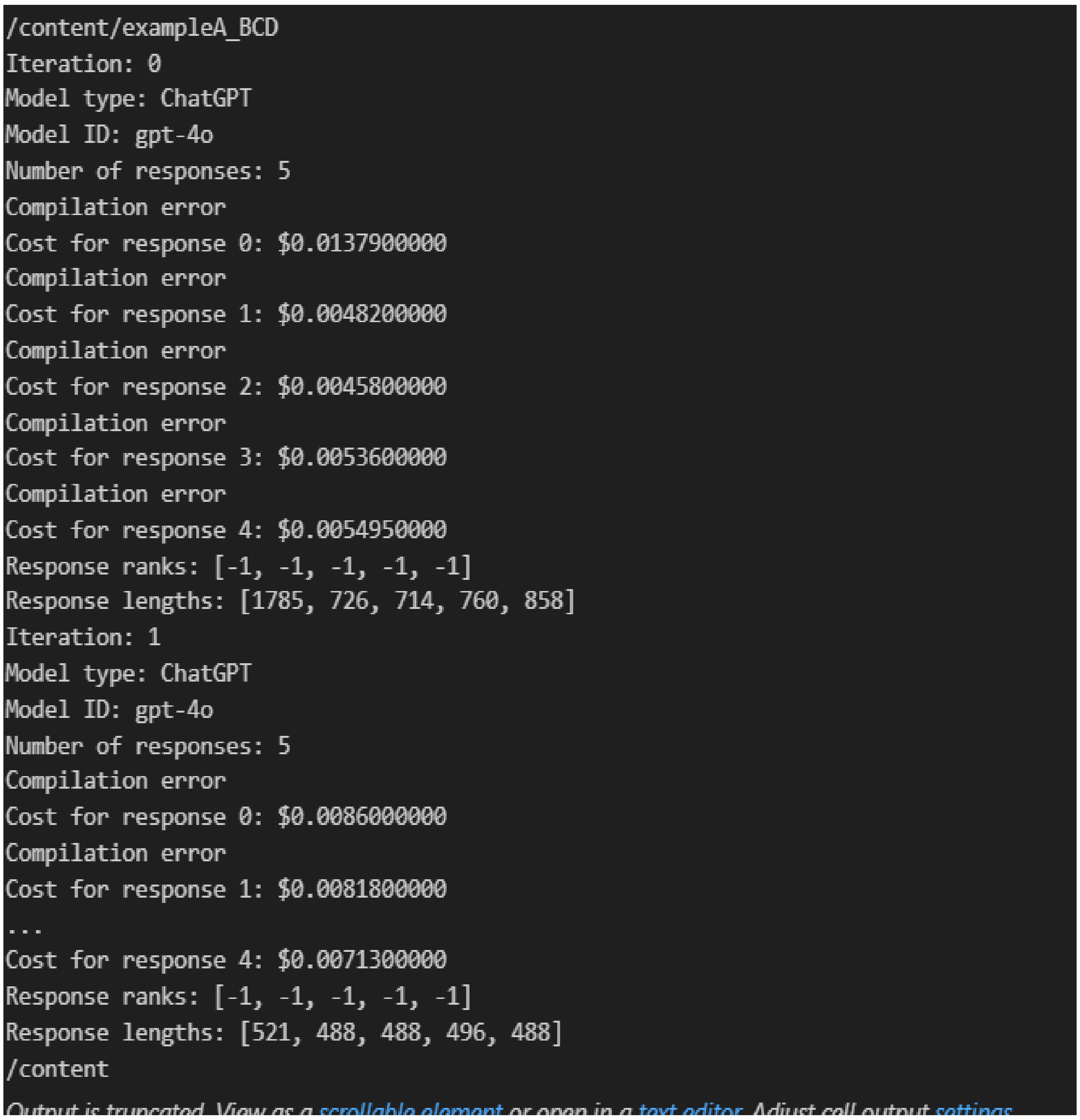


```
/content/exampleA_BCD
Iteration: 0
Model type: ChatGPT
Model ID: gpt-4o-mini
Number of responses: 5
No modules found in response
Cost for response 0: $0.0000523500
No modules found in response
Cost for response 1: $0.0000523500
No modules found in response
Cost for response 2: $0.0000523500
No modules found in response
Cost for response 3: $0.0000523500
No modules found in response
Cost for response 4: $0.0000523500
Response ranks: [-2, -2, -2, -2, -2]
Response lengths: [0, 0, 0, 0, 0]
Iteration: 1
Model type: ChatGPT
Model ID: gpt-4o-mini
Number of responses: 5
No modules found in response
Cost for response 0: $0.0000531000
```

And the first output turns no response.

Second Trial:

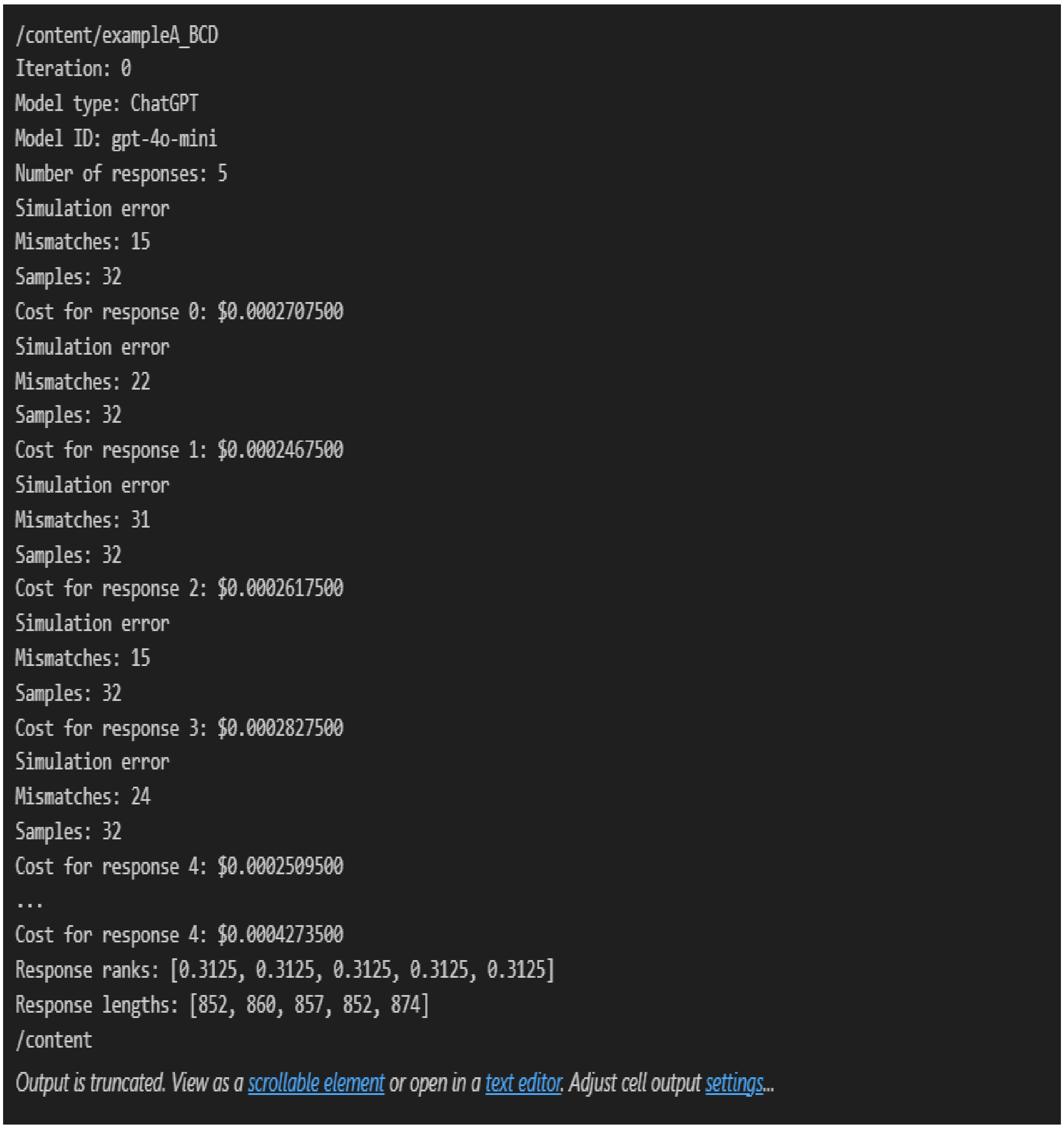
This time i didnt change the prompt a lot but i do encounter a lot of basic compiler issue mostly is because my testbench is not corresponding with the auto chip rule



```
/content/exampleA_BCD
Iteration: 0
Model type: ChatGPT
Model ID: gpt-4o
Number of responses: 5
Compilation error
Cost for response 0: $0.0137900000
Compilation error
Cost for response 1: $0.0048200000
Compilation error
Cost for response 2: $0.0045800000
Compilation error
Cost for response 3: $0.0053600000
Compilation error
Cost for response 4: $0.0054950000
Response ranks: [-1, -1, -1, -1, -1]
Response lengths: [1785, 726, 714, 760, 858]
Iteration: 1
Model type: ChatGPT
Model ID: gpt-4o
Number of responses: 5
Compilation error
Cost for response 0: $0.0086000000
Compilation error
Cost for response 1: $0.0081800000
...
Cost for response 4: $0.0071300000
Response ranks: [-1, -1, -1, -1, -1]
Response lengths: [521, 488, 488, 496, 488]
/content
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

So i change the testbench name from binary_to_bcd_converter to tb and everything works it can run finally.

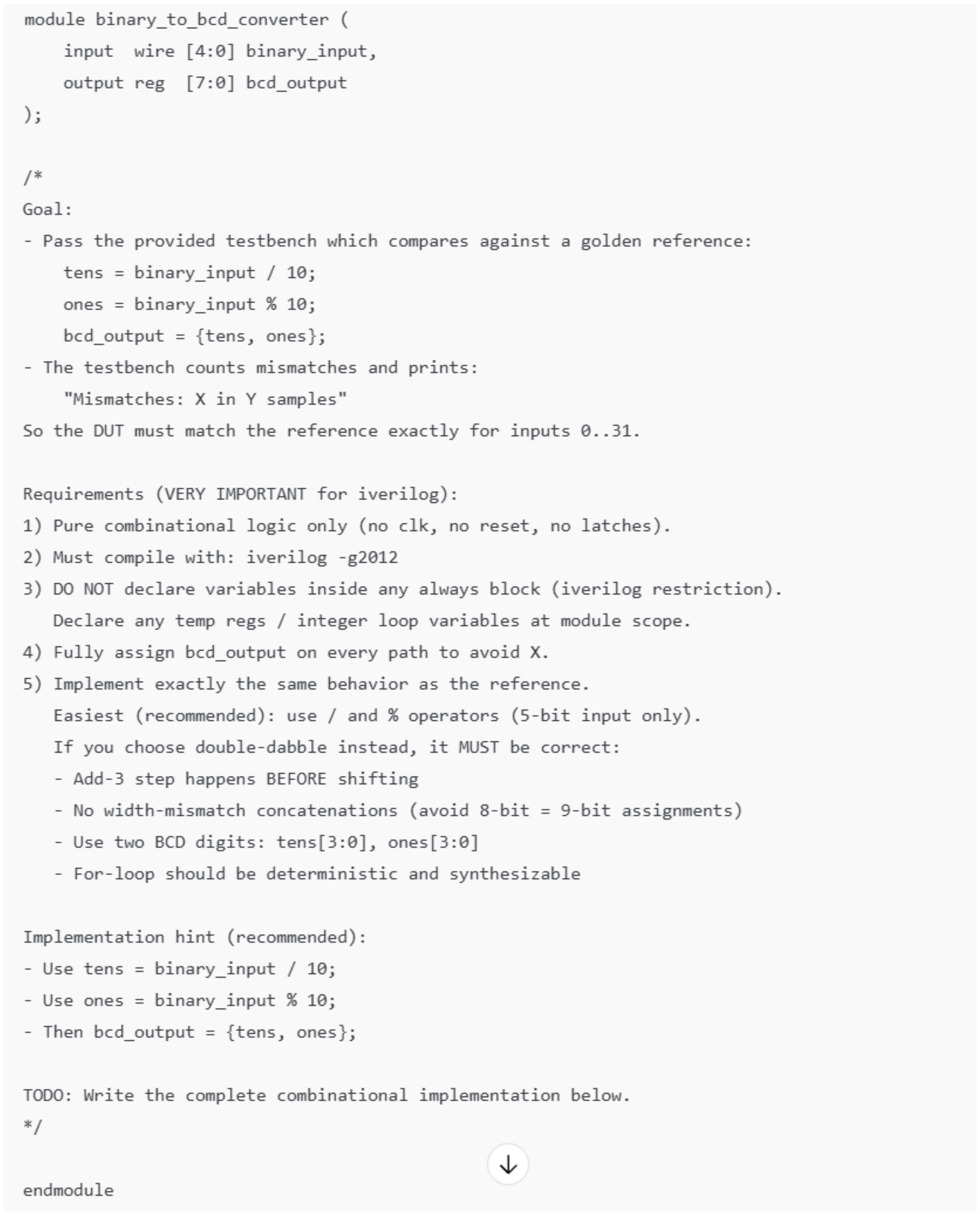
Third Trial:
I still didnt change the prompt a lot but it turns out to be some problems with the simulation result which is not corresponding with the ideal output from the testbench which ideally should be.



```
/content/exampleA_BCD
Iteration: 0
Model type: ChatGPT
Model ID: gpt-4o-mini
Number of responses: 5
Simulation error
Mismatches: 15
Samples: 32
Cost for response 0: $0.0002707500
Simulation error
Mismatches: 22
Samples: 32
Cost for response 1: $0.0002467500
Simulation error
Mismatches: 31
Samples: 32
Cost for response 2: $0.0002617500
Simulation error
Mismatches: 15
Samples: 32
Cost for response 3: $0.0002827500
Simulation error
Mismatches: 24
Samples: 32
Cost for response 4: $0.0002509500
...
Cost for response 4: $0.0004273500
Response ranks: [0.3125, 0.3125, 0.3125, 0.3125, 0.3125]
Response lengths: [852, 860, 857, 852, 874]
/content
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

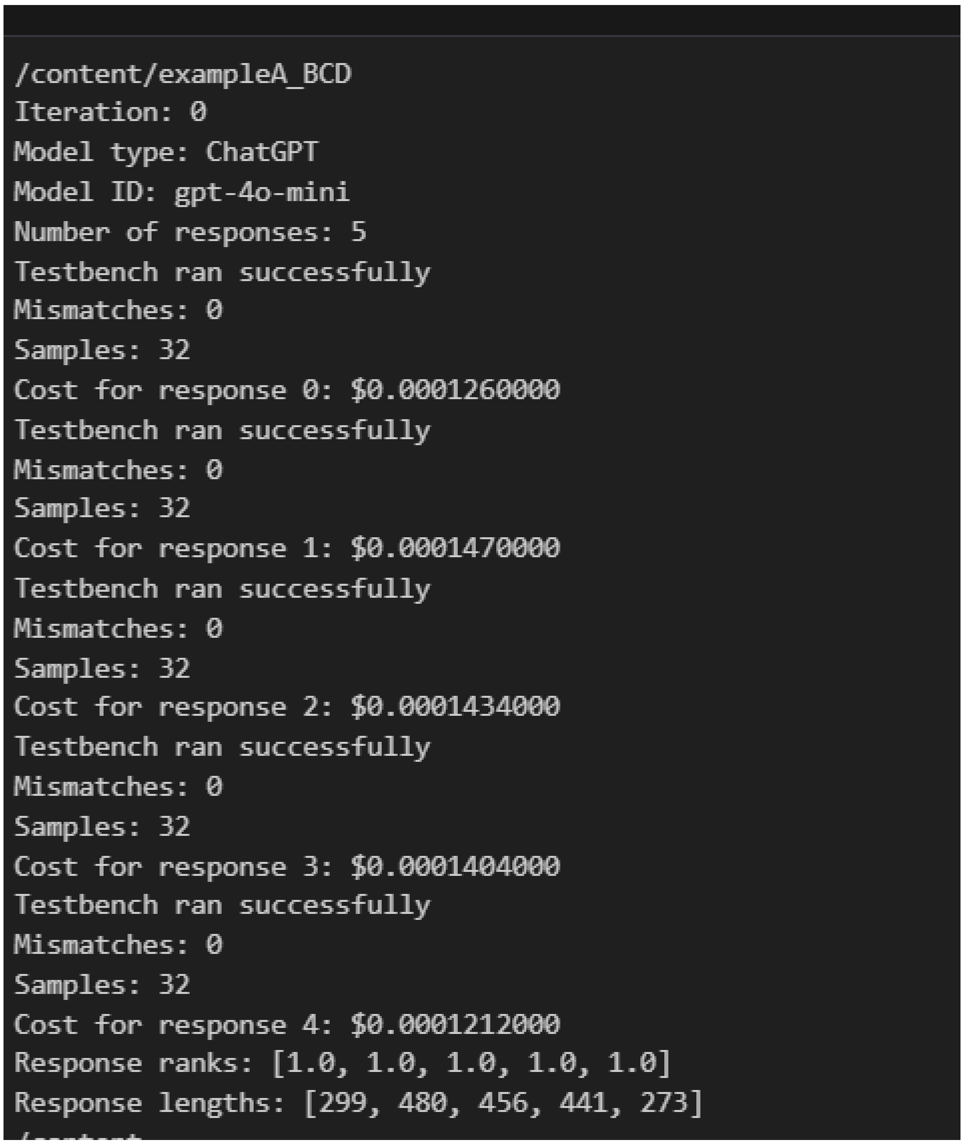
Fourth Trial：

This time i change the prompt to the right version which is : I ONLY told the llm to just using the basic assign language which avoid the complex and hard to realize result.



```
module binary_to_bcd_converter (
    input  wire [4:0] binary_input,
    output reg  [7:0] bcd_output
);

/*
Goal:
- Pass the provided testbench which compares against a golden reference:
    tens = binary_input / 10;
    ones = binary_input % 10;
    bcd_output = {tens, ones};
- The testbench counts mismatches and prints:
    "Mismatches: X in Y samples"
So the DUT must match the reference exactly for inputs 0..31.

Requirements (VERY IMPORTANT for iverilog):
1) Pure combinational logic only (no clk, no reset, no latches).
2) Must compile with: iverilog -g2012
3) DO NOT declare variables inside any always block (iverilog restriction).
   Declare any temp regs / integer loop variables at module scope.
4) Fully assign bcd_output on every path to avoid X.
5) Implement exactly the same behavior as the reference.
   Easiest (recommended): use / and % operators (5-bit input only).
   If you choose double-dabble instead, it MUST be correct:
   - Add-3 step happens BEFORE shifting
   - No width-mismatch concatenations (avoid 8-bit = 9-bit assignments)
   - Use two BCD digits: tens[3:0], ones[3:0]
   - For-loop should be deterministic and synthesizable

Implementation hint (recommended):
- Use tens = binary_input / 10;
- Use ones = binary_input % 10;
- Then bcd_output = {tens, ones};

TODO: Write the complete combinational implementation below.
*/

endmodule
```

And it finally pass the test.



```
/content/exampleA_BCD
Iteration: 0
Model type: ChatGPT
Model ID: gpt-4o-mini
Number of responses: 5
Testbench ran successfully
Mismatches: 0
Samples: 32
Cost for response 0: $0.0001260000
Testbench ran successfully
Mismatches: 0
Samples: 32
Cost for response 1: $0.0001470000
Testbench ran successfully
Mismatches: 0
Samples: 32
Cost for response 2: $0.0001434000
Testbench ran successfully
Mismatches: 0
Samples: 32
Cost for response 3: $0.0001404000
Testbench ran successfully
Mismatches: 0
Samples: 32
Cost for response 4: $0.0001212000
Response ranks: [1.0, 1.0, 1.0, 1.0, 1.0]
Response lengths: [299, 480, 456, 441, 273]
```

The Second Example :

I choose the registor shift. And the initial prompt is shown as follows .

'''

I am trying to create a Verilog model for a shift register. It must meet the following specifications:

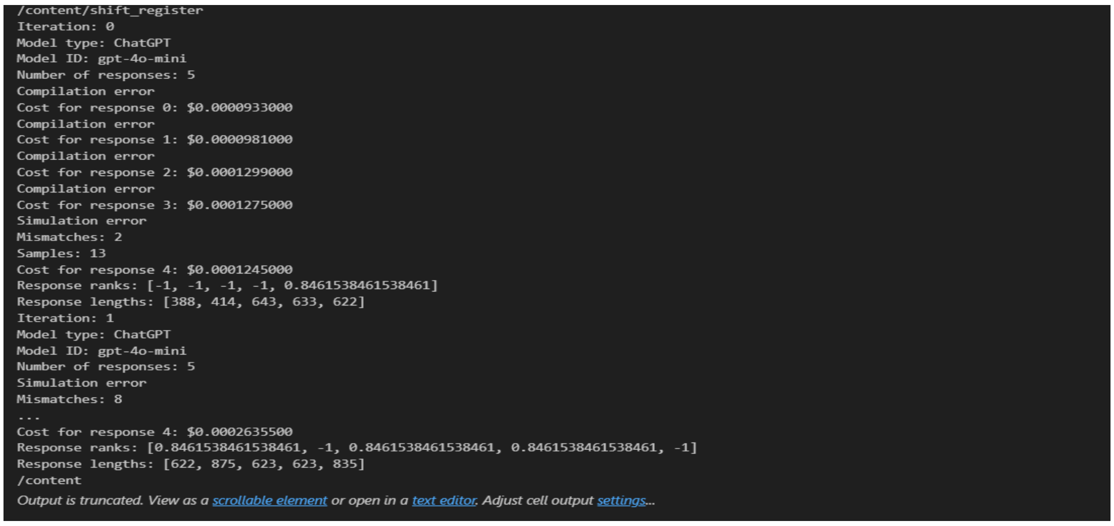
- Inputs:
  - Clock
  - Active-low reset
  - Data (1 bit)
  - Shift enable
- Outputs:
  - Data (8 bits)

How would I write a design that meets these specifications?

```
module shift_register(
            input logic clk,
            input logic reset_n,
            input logic data_in,
            input logic shift_enable,
            output logic [7:0] data_out

);
```

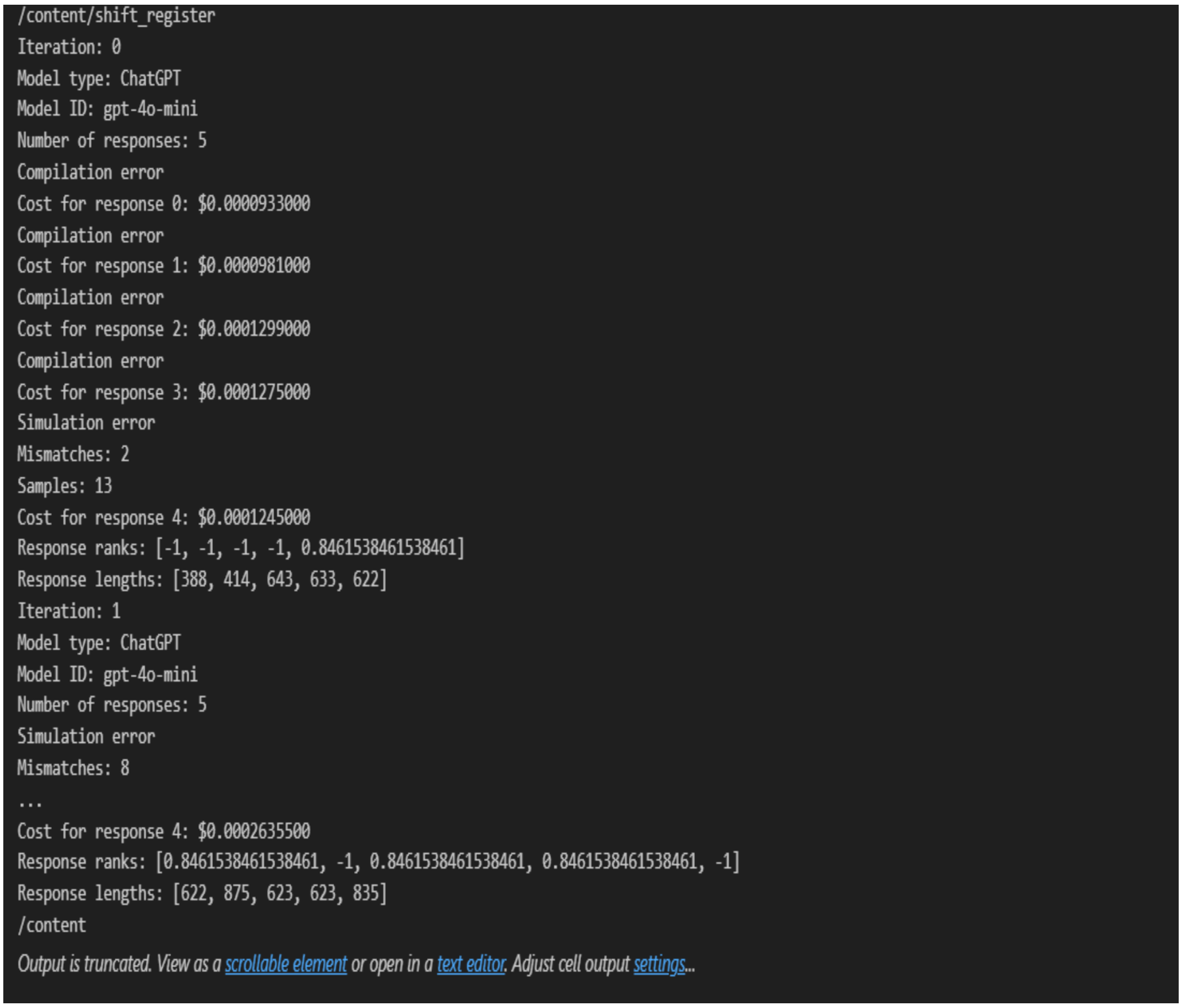
But it pops up that there is a violation which is against the compilation rule, so i return back and double check whether there might be a issue with the code.

After see the log.txt. I found the mistake lies at the testbench. I am not using correct testbench name . So i changed to tb which is corresponding with the name of the Autochip Python logic.



```
/content/shift_register
Iteration: 0
Model type: ChatGPT
Model ID: gpt-4o-mini
Number of responses: 5
Compilation error
Cost for response 0: $0.0000933000
Compilation error
Cost for response 1: $0.0000981000
Compilation error
Cost for response 2: $0.0001299000
Compilation error
Cost for response 3: $0.0001275000
Simulation error
Mismatches: 2
Samples: 13
Cost for response 4: $0.0001245000
Response ranks: [-1, -1, -1, -1, 0.8461538461538461]
Response lengths: [388, 414, 643, 633, 622]
Iteration: 1
Model type: ChatGPT
Model ID: gpt-4o-mini
Number of responses: 5
Simulation error
Mismatches: 8
...
Cost for response 4: $0.0002635500
Response ranks: [0.8461538461538461, -1, 0.8461538461538461, 0.8461538461538461, -1]
Response lengths: [622, 875, 623, 623, 835]
/content
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Second Trial,
The mistake is the sv code i use llm to generate didnt use the negedge reset_n. So i eliminate the negedge clk and only use the posedge clk.



```
/content/shift_register
Iteration: 0
Model type: ChatGPT
Model ID: gpt-4o-mini
Number of responses: 5
Compilation error
Cost for response 0: $0.0000933000
Compilation error
Cost for response 1: $0.0000981000
Compilation error
Cost for response 2: $0.0001299000
Compilation error
Cost for response 3: $0.0001275000
Simulation error
Mismatches: 2
Samples: 13
Cost for response 4: $0.0001245000
Response ranks: [-1, -1, -1, -1, 0.8461538461538461]
Response lengths: [388, 414, 643, 633, 622]
Iteration: 1
Model type: ChatGPT
Model ID: gpt-4o-mini
Number of responses: 5
Simulation error
Mismatches: 8
...
Cost for response 4: $0.0002635500
Response ranks: [0.8461538461538461, -1, 0.8461538461538461, 0.8461538461538461, -1]
Response lengths: [622, 875, 623, 623, 835]
/content
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

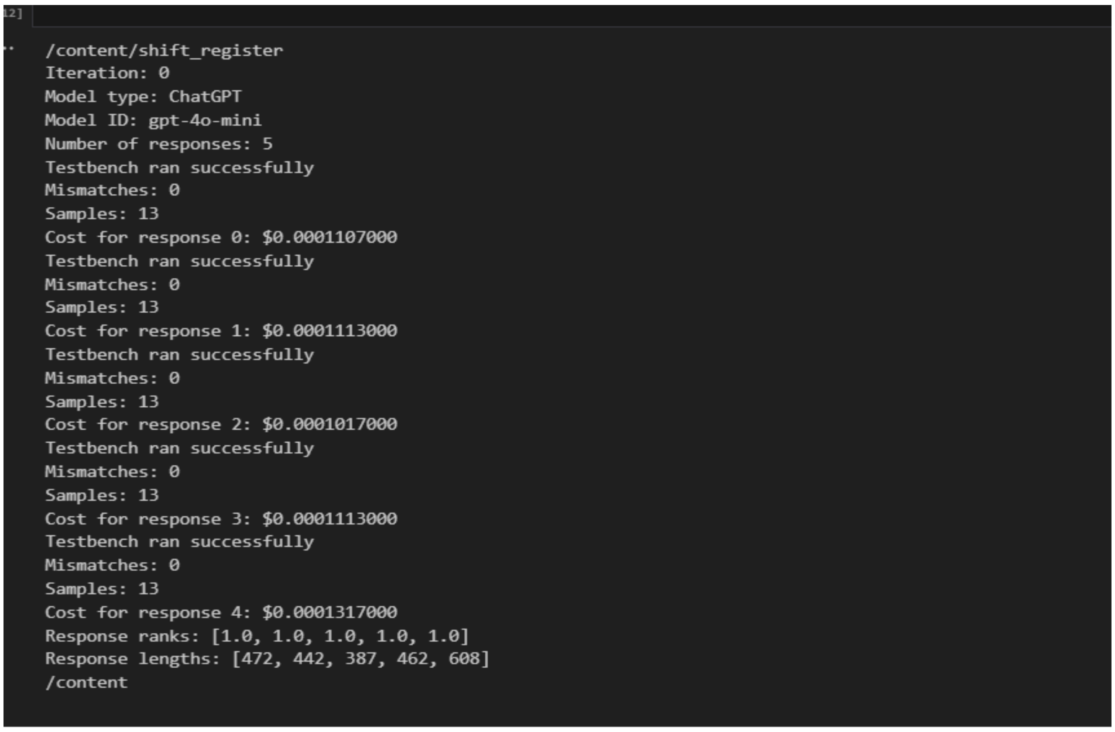
The Third prompt looks like that:
I am trying to create a SystemVerilog model for an 8-bit shift register.

Ports (MUST match exactly):
- input   logic clk
- input   logic reset_n   (active-low SYNCHRONOUS reset)
- input   logic data_in   (1-bit)
- input   logic shift_enable
- output logic [7:0] data_out

Behavior (MUST match):
- On each posedge clk:
  - If reset_n == 0, set data_out <= 8'b0 (synchronous reset: DO NOT use negedge reset in sensitivity list)
  - Else if shift_enable == 1, shift left and insert data_in into LSB:
    data_out <= {data_out[6:0], data_in}
  - Else hold previous data_out (no change)

Constraints:
- Use SystemVerilog syntax (logic, always_ff).
- Single module only: shift_register.
- No additional helper modules.
- Provide ONLY the complete module code.

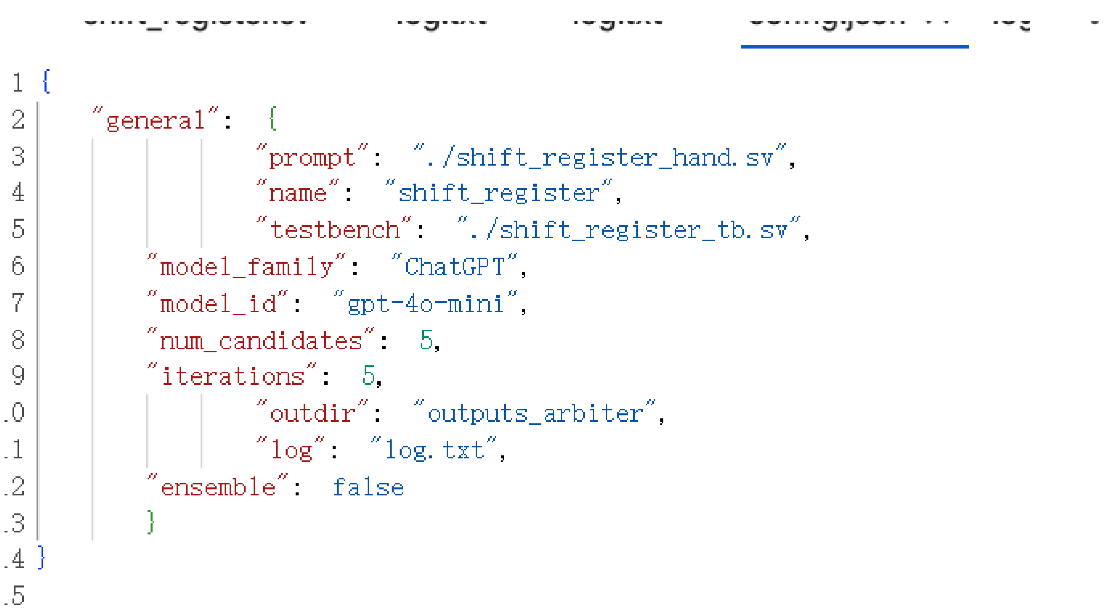
And the test finally pass all the testbench after iteration



```
/content/shift_register
Iteration: 0
Model type: ChatGPT
Model ID: gpt-4o-mini
Number of responses: 5
Testbench ran successfully
Mismatches: 0
Samples: 13
Cost for response 0: $0.0001107000
Testbench ran successfully
Mismatches: 0
Samples: 13
Cost for response 1: $0.0001113000
Testbench ran successfully
Mismatches: 0
Samples: 13
Cost for response 2: $0.0001017000
Testbench ran successfully
Mismatches: 0
Samples: 13
Cost for response 3: $0.0001113000
Testbench ran successfully
Mismatches: 0
Samples: 13
Cost for response 4: $0.0001317000
Response ranks: [1.0, 1.0, 1.0, 1.0, 1.0]
Response lengths: [472, 442, 387, 462, 608]
/content
```

For part I(b):

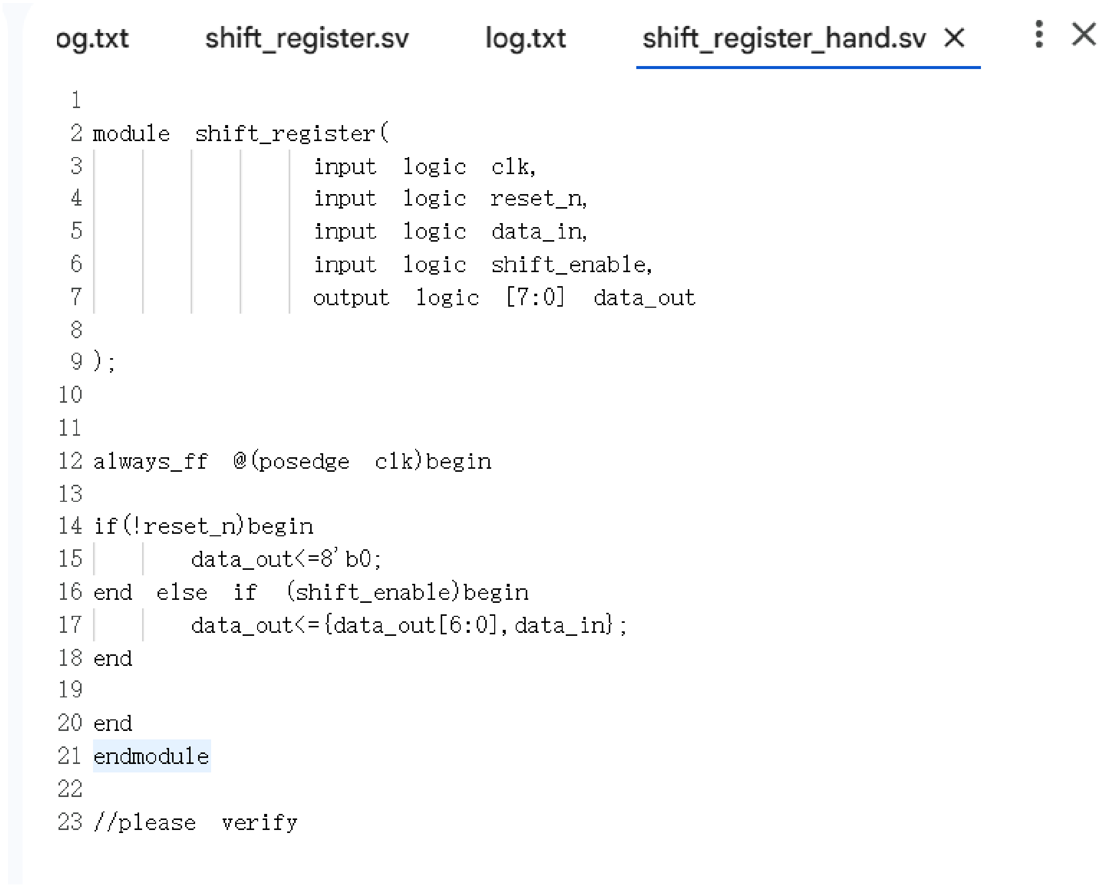
I changed the config.json file



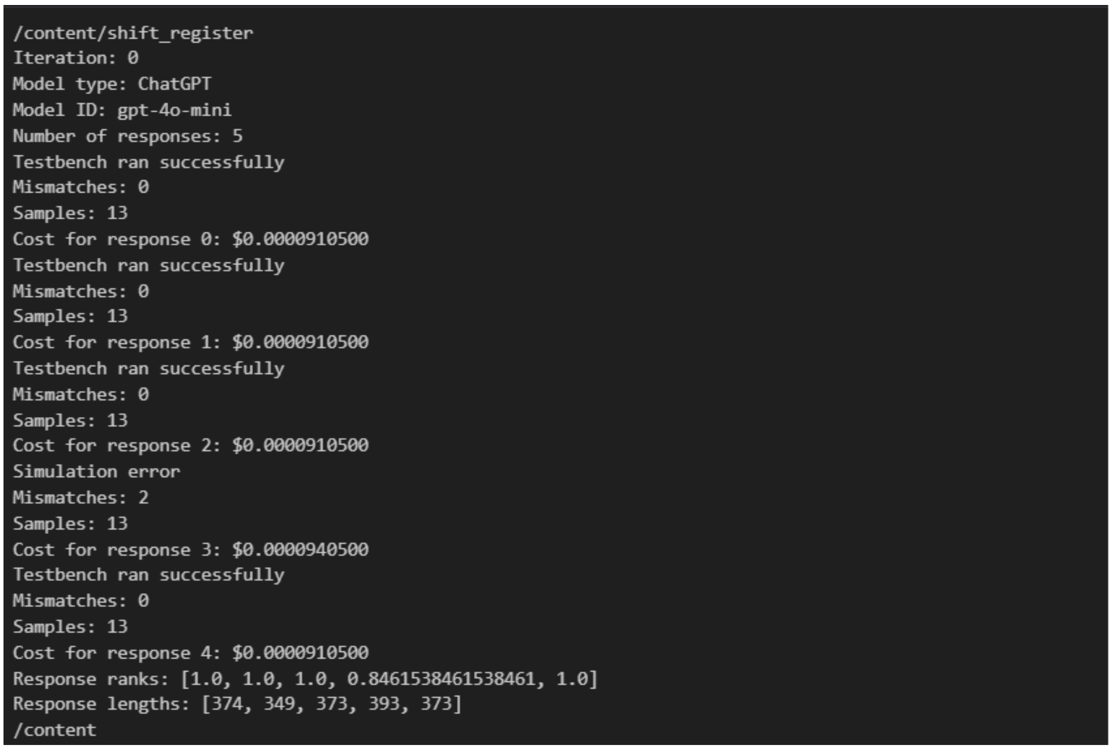
```
{
    "general":  {
                "prompt":  "./shift_register_hand.sv",
                "name":  "shift_register",
                "testbench":  "./shift_register_tb.sv",
        "model_family":  "ChatGPT",
        "model_id":  "gpt-4o-mini",
        "num_candidates":  5,
        "iterations":  5,
                "outdir":  "outputs_arbiter",
                "log":  "log.txt",
        "ensemble":  false
        }
}
```

Change the promt from shift_register.sv to shift_register_hand.sv and pass the test

This is the hand written version.



```

module  shift_register(
                input  logic  clk,
                input  logic  reset_n,
                input  logic  data_in,
                input  logic  shift_enable,
                output  logic  [7:0]  data_out

);


always_ff  @(posedge  clk)begin

if(!reset_n)begin
        data_out<=8'b0;
end  else  if  (shift_enable)begin
        data_out<={data_out[6:0],data_in};
end

end
endmodule

//please  verify
```



```
/content/shift_register
Iteration: 0
Model type: ChatGPT
Model ID: gpt-4o-mini
Number of responses: 5
Testbench ran successfully
Mismatches: 0
Samples: 13
Cost for response 0: $0.0000910500
Testbench ran successfully
Mismatches: 0
Samples: 13
Cost for response 1: $0.0000910500
Testbench ran successfully
Mismatches: 0
Samples: 13
Cost for response 2: $0.0000910500
Simulation error
Mismatches: 2
Samples: 13
Cost for response 3: $0.0000940500
Testbench ran successfully
Mismatches: 0
Samples: 13
Cost for response 4: $0.0000910500
Response ranks: [1.0, 1.0, 1.0, 0.8461538461538461, 1.0]
Response lengths: [374, 349, 373, 393, 373]
/content
```