Example 1: Binary to Binary-coded-Decimal

Prompt:

```
// make a Verilog design code for binary to binary-coded-decimal number, it takes 5 input binary //number
//and outputs a 8 bit number of which lower 4 are of ones's place
//and upper 4 are ten's place
//I am giving you module definition use that
// don't change module definition

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

Iteration	Response	Status	Mismatches	Samples	Cost ($)	Rank	Length
0	0	No module found	-	-	0.0000306	-2	0
0	1	No module found	-	-	0.0000306	-2	0
0	2	No module found	-	-	0.0000306	-2	0
0	3	No module found	-	-	0.0000306	-2	0
0	4	No module found	-	-	0.0000306	-2	0
1	0	Simulation error	76	163	0.0001724	0.534	793
1	1	Simulation error	115	163	0.0001544	0.294	668
1	2	Simulation error	76	163	0.0001802	0.534	796
1	3	Simulation error	155	163	0.0001748	0.049	746
1	4	Success	0	163	0.0000980	1.0	390

Code:

```
module top_module (
   input [4:0] binary_input,
   output reg [7:0] bcd_output
);
   integer binary;
   reg [3:0] ones;
   reg [3:0] tens;

   always @* begin
      // Convert binary input to decimal
      binary = binary_input;
      ones = binary % 10;
      tens = binary / 10;

      // Assign BCD output
      bcd_output = {tens, ones};
   end
endmodule
```

Comment: Although using %,/ operators are expensive for hardware and it would be better if it had used bit shift and add method. I would have to mention that it in the prompt but then never got a correct code even after increasing iterations to 10.

Example 2: LFSR

Prompt:

Design an 8-bit LFSR in Verilog:
It generates a sequence of bits by shifting the bits left in a shift register and using a linear function, typically XOR gates, to combine specific bits (taps) to produce the next bit.

```
module top_module (
    input clk,
    input reset_n,
    output [7:0] data
);
```

reset value when reset_n is low (initial state): 8'b10001010
taps at data[0], data[3], data[5], data[6]

Iteration	**Response**	**Status**	**Mismatches**	**Samples**	**Cost ($)**	**Rank**	**Length**
0	0	Success	0	565	0.0001113	1.0	465
0	1	Compilation Error	-	-	0.0001125	-1	474
0	2	Success	0	565	0.0001095	1.0	397
0	3	Success	0	565	0.0001107	1.0	427
0	4	Compilation Error	-	-	0.0001251	-1	579

Comment: The prompt was quite sufficient for this level of design. I had to refine prompt it kept failing through all 10 iterations.
Code:

```
module top_module (
    input clk,
    input reset_n,
    output reg [7:0] data
);

always @(posedge clk or negedge reset_n) begin
    if (!reset_n) begin
        data <= 8'b10001010; // Reset value
    end else begin
        // Calculate the next bit using taps at data[0], data[3], data[5], and data[6]
        data <= {data[6:0], data[0] ^ data[3] ^ data[5] ^ data[6]};
    end
end
endmodule
```

Example 3: Shift Register

Prompt:

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)

here is an starting module for you:

```
module top_module(
  input clk,
  input reset_n,
  input data_n,
  input shift_enable,
  output [7:0] data_out
);
```

Iteration	Response	Status	Mismatches	Samples	Cost ($)	Rank	Length
0	0	Compilation Error	-	-	0.0000902	-1	322
0	1	Compilation Error	-	-	0.0000986	-1	367
0	2	Compilation Error	-	-	0.0001022	-1	396
0	3	Compilation Error	-	-	0.0000986	-1	366
0	4	Compilation Error	-	-	0.0000890	-1	308
1	0	Compilation Error	-	-	0.0001652	-1	345
1	1	Simulation Error	27	65	0.0001598	0.5846	310
1	2	Simulation Error	27	65	0.0001598	0.5846	310
1	3	Simulation Error	27	65	0.0001718	0.5846	383
1	4	Simulation Error	27	65	0.0001736	0.5846	384
2	0	Success	0	65	0.0001674	1.0	350
2	1	Success	0	65	0.0001662	1.0	343
2	2	Success	0	65	0.0001650	1.0	339
2	3	Simulation Error	27	65	0.0001650	0.5846	329
2	4	Success	0	65	0.0001662	1.0	343

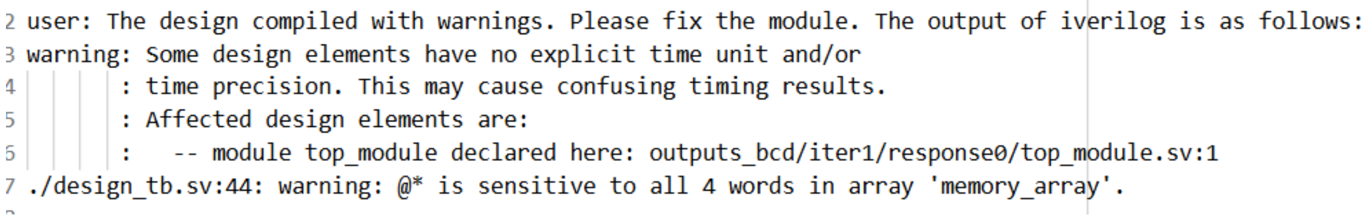
Observation:

For an example where I needed three modules, the autochip framework was not able to extract all code.

It also keeps showing compilation warning which can be ignored. For example:

always @(*)

this was giving compilation warning but that was that was the best way to write the code.



```
user: The design compiled with warnings. Please fix the module. The output of iverilog is as follows:
warning: Some design elements have no explicit time unit and/or
       : time precision. This may cause confusing timing results.
       : Affected design elements are:
       :   -- module top_module declared here: outputs_bcd/iter1/response0/top_module.sv:1
./design_tb.sv:44: warning: @* is sensitive to all 4 words in array 'memory_array'.
```

Sometimes if the prompt is not informative enough, response doesn’t improve at any iteration. Refining prompt and human intervention is needed.