

VeriThoughts Inconsistent Example Report - Binary to BCD (Failed Case)

Name: Naveen Kumar Senthil Kumar

Netid: ns6503

Ground Truth

```
module bin_to_bcd #(parameter WIDTH = 5) (  
    input wire [WIDTH-1:0] bin,  
    output reg [3:0] bcd_tens,  
    output reg [3:0] bcd_ones  
);  
    reg [5:0] tmp;  
    always @* begin  
        if (bin >= 30) begin bcd_tens = 4'd3; tmp = bin - 30; end  
        else if (bin >= 20) begin bcd_tens = 4'd2; tmp = bin - 20; end  
        else if (bin >= 10) begin bcd_tens = 4'd1; tmp = bin - 10; end  
        else begin bcd_tens = 4'd0; tmp = bin; end  
        bcd_ones = tmp[3:0];  
        if (tmp >= 10) bcd_ones = tmp - 10;  
    end  
endmodule
```

Question (Prompt)

Write a Verilog module that takes a 5-bit binary input and outputs the number of set bits in BCD format.

This prompt defines a different problem, counting the number of 1s (Hamming weight), not converting a binary number to BCD. Hence, the generated code does not correspond to the ground truth logic.

Generated Verilog

```
module bin_to_bcd #(parameter WIDTH = 5) (  
    input [WIDTH-1:0] bin,  
    output reg [3:0] bcd  
);  
    integer i;  
    always @* begin  
        bcd = 0;  
        for (i = 0; i < WIDTH; i = i + 1) begin  
            if (bin[i]) bcd = bcd + 1;  
        end  
    end  
endmodule
```

```
    end
end
endmodule
```

This code performs a bit-count operation instead of a binary-to-decimal conversion. It produces only one BCD output (`bcd`), not two digits (`bcd_tens`, `bcd_ones`). Therefore, it's functionally inconsistent with the ground truth.

Verification Result

verified: false

Reason: Logical mismatch between ground_truth and generated_verilog due to prompt inconsistency.

Runtime Warning Observed

[rank0]:[W1016 01:44:37.529344942 ProcessGroupNCCL.cpp:1250] Warning: WARNING: process group has NOT been destroyed before we destruct ProcessGroupNCCL. On normal program exit, the application should call destroy_process_group to ensure that any pending NCCL operations have finished in this process. In rare cases this process can exit before this point and block the progress of another member of the process group. This constraint has always been present, but this warning has only been added since PyTorch 2.4 (function operator())

This message indicates that the NCCL communication group used in distributed PyTorch processes was not explicitly destroyed before program termination. It's unrelated to the Verilog logic itself but occurred because the distributed evaluation script exited before calling torch.distributed.destroy_process_group(). The fix is to call this cleanup function in a finally block to safely terminate NCCL processes.

Inference

The inconsistency lies in the question/prompt, not the ground truth or generated code syntax. The model followed the wrong instruction faithfully, producing code that matches the prompt but not the reference design. This highlights the importance of prompt alignment in Verilog code synthesis tasks, precise functional wording ensures correct generation and verification.

Comparison with the Correct Prompt

Correct Prompt: Write a synthesizable Verilog module that converts a 5-bit unsigned binary input (0..31) to two BCD digits (tens, ones).

Aspect	Inconsistent Prompt	Correct Prompt
Intent	Count number of set bits	Convert binary number (0–31) into two BCD digits
Outputs	One BCD nibble (bcd)	Two 4-bit outputs (bcd_tens, bcd_ones)

Logic	Hamming weight loop	Arithmetic conversion (/10 or subtractive)
Functional Goal	Bit-counting	Decimal representation
Verification	Failed (different logic)	Passed (aligned with ground truth)

Observation:

Root Cause: Incorrect problem statement in the prompt.

Effect: Generated code solved a different task to verification failed.

Fix: Align the prompt to specify conversion of binary number (0..31) to two BCD digits (tens, ones).

Additional Note: Ignore NCCL warning or add cleanup code, it's a runtime artifact unrelated to HDL logic validation.

Fixed prompt:

```
{
  "ground_truth": "module bin_to_bcd #(parameter WIDTH = 5) (\n  input wire [WIDTH-1:0] bin,\n  output reg [3:0] bcd_tens,\n  output reg [3:0] bcd_ones\n);\n  // Valid for WIDTH=5 (0..31). Simple arithmetic mapping to BCD.\n  reg [5:0] tmp;\n  always @* begin\n    if (bin >= 30) begin bcd_tens = 4'd3; tmp = bin - 30; end\n    else if (bin >= 20) begin bcd_tens = 4'd2; tmp = bin - 20; end\n    else if (bin >= 10) begin bcd_tens = 4'd1; tmp = bin - 10; end\n    else begin bcd_tens = 4'd0; tmp = bin; end\n    bcd_ones = tmp[3:0];\n    if (tmp >= 10) bcd_ones = tmp - 10; // safety, shouldn't trigger for 0..31\n  end\nendmodule",
  "question": "Write a synthesizable Verilog module that converts a 5-bit unsigned binary input (0..31) to two BCD digits (tens, ones).",
  "generated_verilog": "module bin_to_bcd #(parameter WIDTH = 5) (\n  input wire [WIDTH-1:0] bin,\n  output reg [3:0] bcd_tens,\n  output reg [3:0] bcd_ones\n);\n  reg [5:0] tmp;\n  always @* begin\n    if (bin >= 30) begin bcd_tens = 4'd3; tmp = bin - 30; end\n    else if (bin >= 20) begin bcd_tens = 4'd2; tmp = bin - 20; end\n    else if (bin >= 10) begin bcd_tens = 4'd1; tmp = bin - 10; end\n    else begin bcd_tens = 4'd0; tmp = bin; end\n    bcd_ones = tmp[3:0];\n    if (tmp >= 10) bcd_ones = tmp - 10;\n  end\nendmodule",
  "verified": true
}
```