Skylar Stockham
P.E Gaillardon

LLM Trial Logs

The following logs record our outputs from RapidGPTs. The benchmark prompts are colored in green, then the response from each run is pasted in and numbered.

Universal Prompt

Benchmark 1:

Can you heat me a solution and then mix it with another?

```
module heat_and_mix ( input wire a_fluid, input wire b_fluid, output wire out_fluid );

 // Heat the solution heat_25px_0 heat (.in_fluid(a_fluid), .out_fluid(heated_fluid)); // Mix the heated solution with another solution

diffmix_25px_0 mix (.a_fluid(heated_fluid), .b_fluid(b_fluid), .out_fluid(out_fluid));

endmodule
```

Success

2

```
module lab_chip_design(
  input wire soln1,
  input wire soln2,
  output wire connect0
);

  // Heat the solution
  heat_25px_0 heat0 (.in_fluid(soln1), .out_fluid(soln1_heat));

  // Mix the heated solution with another solution
  diffmix_25px_0 mix0 (.a_fluid(soln1_heat), .b_fluid(soln2), .out_fluid(connect0));

Endmodule
```

3

```
module lab_chip_design(
  input wire soln1,
  input wire soln2,
  output wire connect0
);

  // Heat the solution
  heat_25px_0 heat0 (.in_fluid(soln1), .out_fluid(soln1_heat));
  // Mix the heated solution with another solution
  diffmix_25px_0 mix0 (.a_fluid(soln1_heat), .b_fluid(soln2), .out_fluid(connect0));

Endmodule
```

4

```
module heat_and_mix (
  input wire a_fluid,
  input wire b_fluid,
  output wire out_fluid
);

  // Heat the solution
  heat_25px_0 heat (.in_fluid(a_fluid), .out_fluid(heated_fluid));

  // Mix the heated solution with another solution
  diffmix_25px_0 mix (.a_fluid(heated_fluid), .b_fluid(b_fluid), .out_fluid(out_fluid));

Endmodule
```

5

```
module heat_and_mix (
  input wire a_fluid,
  input wire b_fluid,
  output wire out_fluid
);

  // Heat the first solution
  heat_25px_0 heat1 (.in_fluid(a_fluid), .out_fluid(heated_a));

  // Mix the heated solution with the second solution
  diffmix_25px_0 mix (.a_fluid(heated_a), .b_fluid(b_fluid), .out_fluid(out_fluid));

Endmodule
```

5/5

Mix 2 solutions, then heat them

```
module lab_chip (
  input wire soln1,
  input wire soln2,
  output wire heated_soln
);

  // Mixer component
  diffmix_25px_0 mix0 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(mix_out));

  // Heating component
  heating_component heat0 (.in_fluid(mix_out), .out_fluid(heated_soln));

endmodule
```

2

```
module lab_chip(
  input wire soln1,
  input wire soln2,
  output wire heat_out
);

  // Mixer component
  diffmix_25px_0 mix0 (
    .a_fluid(soln1),
    .b_fluid(soln2),
    .out_fluid(mix_out)
  );

  // Heating component
  heat_0 heat (
    .in_fluid(mix_out),
    .out_fluid(heat_out)
  );

Endmodule
```

3

```
module lab_chip (
  input wire soln1,
  input wire soln2,
  output wire heat_out
);

  wire mix_out;

  diffmix_25px_0 mix0 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(mix_out));
  heat_module heat0 (.in_fluid(mix_out), .out_fluid(heat_out));

endmodule

module heat_module (
  input wire in_fluid,
  output wire out_fluid
);

  // Add your heat logic here

Endmodule
```

FAIL. We can see here that the Program did not listen to our explicit demands.

4

```
// Mixer component
diffmix_25px_0 mix0 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(mix_out));

// Heating component
heating_component heat0 (.in_fluid(mix_out), .out_fluid(heated_out));
```

FAIL. Program failed to generate all the supporting verilog

5

```
module lab_chip(
  input wire soln1,
  input wire soln2,
  output wire connect0
);

  // Mixer component
  diffmix_25px_0 mix0 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(mix_out));

  // Heater component
  heater_50px_0 heat0 (.in_fluid(mix_out), .out_fluid(connect0));

Endmodule
```

mix three solutions

1

```
module mix_three_solutions (
  input wire a_fluid,
  input wire b_fluid,
  input wire c_fluid,
  output wire out_fluid
);

  diffmix_25px_0 mix0 (.a_fluid(a_fluid), .b_fluid(b_fluid), .out_fluid(connect0));
  diffmix_25px_1 mix1 (.a_fluid(connect0), .b_fluid(c_fluid), .out_fluid(out_fluid));

Endmodule
```

2

```
module mix_three_solutions (
  input wire a_fluid,
  input wire b_fluid,
  input wire c_fluid,
  output wire out_fluid
);
```

```
  diffmix_25px_0 mix0 (.a_fluid(a_fluid), .b_fluid(b_fluid), .out_fluid(connect0));
  diffmix_25px_1 mix1 (.a_fluid(connect0), .b_fluid(c_fluid), .out_fluid(out_fluid));

Endmodule
```

3

```
module mix_three_solutions (
  input soln1,
  input soln2,
  input soln3,
  output mixed_soln
);

  wire mix1_out;
  wire mix2_out;

  diffmix_25px_0 mix1 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(mix1_out));
  diffmix_25px_1 mix2 (.a_fluid(mix1_out), .b_fluid(soln3), .out_fluid(mix2_out));

  assign mixed_soln = mix2_out;

Endmodule
```

4

```
module mix_three_solutions (
  input wire a_fluid,
  input wire b_fluid,
  input wire c_fluid,
  output wire out_fluid
);

  wire mix1_out;
  wire mix2_out;

  diffmix_25px_0 mix1 (.a_fluid(a_fluid), .b_fluid(b_fluid), .out_fluid(mix1_out));
  diffmix_25px_0 mix2 (.a_fluid(mix1_out), .b_fluid(c_fluid), .out_fluid(mix2_out));

  assign out_fluid = mix2_out;

Endmodule
```

5

```
module mix_three_solutions (
  input wire soln1,
  input wire soln2,
  input wire soln3,
  output wire mixed_soln
);

  // Mixer instances
  diffmix_25px_0 mix1 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(temp_soln));
  diffmix_25px_1 mix2 (.a_fluid(temp_soln), .b_fluid(soln3), .out_fluid(mixed_soln));

  // Internal wire
  wire temp_soln;

Endmodule
```

Mix 4 solutions, sequentially

```
module sequential_mixer (
  input wire [3:0] soln,
  output wire out_fluid
);

  diffmix_25px_0 mix0 (.a_fluid(soln[0]), .b_fluid(soln[1]), .out_fluid(temp0));
  diffmix_25px_0 mix1 (.a_fluid(temp0), .b_fluid(soln[2]), .out_fluid(temp1));
  diffmix_25px_0 mix2 (.a_fluid(temp1), .b_fluid(soln[3]), .out_fluid(out_fluid));

Endmodule
```

FAIL.
The code did not follow the input example
2

```
module sequential_mixer (
  input [3:0] soln,
  output wire out_soln
);

  diffmix_25px_0 mix0 (.a_fluid(soln[0]), .b_fluid(soln[1]), .out_fluid(wire1));
  diffmix_25px_0 mix1 (.a_fluid(wire1), .b_fluid(soln[2]), .out_fluid(wire2));
  diffmix_25px_0 mix2 (.a_fluid(wire2), .b_fluid(soln[3]), .out_fluid(out_soln));

Endmodule
```

Fail

I ran it the rest of the time, it failed for each but not explicitly. Our tool is not built to handle multiple bit representations, as it is structural verilog.

3

```
module sequential_mixer (
  input wire [7:0] soln1,
  input wire [7:0] soln2,
  input wire [7:0] soln3,
  input wire [7:0] soln4,
  output wire [7:0] mixed_soln
);
```

```
  wire [7:0] mix1_out;
  wire [7:0] mix2_out;
  wire [7:0] mix3_out;

  diffmix_25px_0 mix1 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(mix1_out));
  diffmix_25px_0 mix2 (.a_fluid(mix1_out), .b_fluid(soln3), .out_fluid(mix2_out));
  diffmix_25px_0 mix3 (.a_fluid(mix2_out), .b_fluid(soln4), .out_fluid(mix3_out));

  assign mixed_soln = mix3_out;

Endmodule
```

4

```
module sequential_mix (
  input wire [7:0] soln1,
  input wire [7:0] soln2,
  input wire [7:0] soln3,
  input wire [7:0] soln4,
  output wire [7:0] mixed_soln
);

  wire [7:0] mix1_out;
  wire [7:0] mix2_out;
  wire [7:0] mix3_out;

  diffmix_25px_0 mix1 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(mix1_out));
  diffmix_25px_0 mix2 (.a_fluid(mix1_out), .b_fluid(soln3), .out_fluid(mix2_out));
  diffmix_25px_0 mix3 (.a_fluid(mix2_out), .b_fluid(soln4), .out_fluid(mix3_out));

  assign mixed_soln = mix3_out;

Endmodule
```

5

```
module sequential_mixer(
  input [3:0] soln,
  output reg out_fluid
);

  wire [3:0] mix_out;

  diffmix_25px_0 mix0 (.a_fluid(soln[0]), .b_fluid(soln[1]), .out_fluid(mix_out[0]));
  diffmix_25px_0 mix1 (.a_fluid(mix_out[0]), .b_fluid(soln[2]), .out_fluid(mix_out[1]));
  diffmix_25px_0 mix2 (.a_fluid(mix_out[1]), .b_fluid(soln[3]), .out_fluid(mix_out[2]));

  assign out_fluid = mix_out[2];

endmodule
```

Mix two solutions, heat them, then mix them sequentially with another two solutions

1

```
module lab_chip(
  input wire soln1,
  input wire soln2,
  input wire soln3,
  input wire soln4,
  output wire result
);

  // Mixer 1
  diffmix_25px_0 mix1 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(mix1_out));

  // Heater
  heater_50px_0 heat1 (.in_fluid(mix1_out), .out_fluid(heat1_out));

  // Mixer 2
  diffmix_25px_0 mix2 (.a_fluid(soln3), .b_fluid(soln4), .out_fluid(mix2_out));

  // Final Mixer
  diffmix_25px_0 mix3 (.a_fluid(heat1_out), .b_fluid(mix2_out), .out_fluid(result));

Endmodule
```

Technically not sequential, but still roughly correctly connected. Lets see how it varies

2

```
module chip_design (
  input wire soln1,
  input wire soln2,
  input wire soln3,
  input wire soln4,
  output wire final_output
);

  // Mixer 1
  diffmix_25px_0 mix1 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(mix1_out));

  // Heater
  heater_50px_0 heat (.in_fluid(mix1_out), .out_fluid(heated_out));

  // Mixer 2
  diffmix_25px_0 mix2 (.a_fluid(soln3), .b_fluid(soln4), .out_fluid(mix2_out));

  // Final Mixer
  diffmix_25px_0 mix3 (.a_fluid(heated_out), .b_fluid(mix2_out), .out_fluid(final_output));

Endmodule
```

3

```
module chip_design(
  input wire soln1,
  input wire soln2,
  input wire soln3,
  input wire soln4,
  output wire final_output
);

  // Mixer 1
  diffmix_25px_0 mix1 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(mix1_out));

  // Heater
  heater_50px_0 heat (.in_fluid(mix1_out), .out_fluid(heated_out));

  // Mixer 2
  diffmix_25px_0 mix2 (.a_fluid(soln3), .b_fluid(soln4), .out_fluid(mix2_out));

  // Final Mixer
  diffmix_25px_0 final_mix (.a_fluid(heated_out), .b_fluid(mix2_out), .out_fluid(final_output));

Endmodule
```

4

```
module chip_design (
  input wire soln1,
  input wire soln2,
  input wire soln3,
  input wire soln4,
  output wire final_output
);

  // Mixer components
  diffmix_25px_0 mix1 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(mix1_out));
  diffmix_25px_1 mix2 (.a_fluid(soln3), .b_fluid(soln4), .out_fluid(mix2_out));

  // Heater component
  heater_50px_0 heat (.in_fluid(mix1_out), .out_fluid(heated_out));

  // Sequential mixer component
  diffmix_25px_2 mix3 (.a_fluid(heated_out), .b_fluid(mix2_out), .out_fluid(final_output));
```

fail , diffmix indexes

```
Endmodule
```

5

```
module chip_design (
  input wire soln1,
  input wire soln2,
  input wire soln3,
  input wire soln4,
  output wire final_output
);

  // Mixer components
  diffmix_25px_0 mix1 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(mix_out1));
  diffmix_25px_1 mix2 (.a_fluid(soln3), .b_fluid(soln4), .out_fluid(mix_out2));

  // Heater component
  heater_50px_0 heat (.in_fluid(mix_out1), .out_fluid(heated_out));

  // Sequential mixer component
  diffmix_25px_2 mix3 (.a_fluid(heated_out), .b_fluid(mix_out2), .out_fluid(final_output));

Endmodule
```

Now we are going to dive into part wise benchmarks

Mix two parts of solutionA with one part of solution B

```
module chip_design(
  input wire soln1,
  input wire soln2,
  output wire connect0
);

  // Mixer 1
  diffmix_25px_0 mix0 (.a_fluid(soln1), .b_fluid(soln1), .out_fluid(temp0));

  // Mixer 2
  diffmix_25px_1 mix1 (.a_fluid(soln2), .b_fluid(soln2), .out_fluid(temp1));

  // Mixer 3
  diffmix_25px_2 mix2 (.a_fluid(temp0), .b_fluid(temp1), .out_fluid(connect0));

Endmodule
```

Fail. Program failed to recognize our serpentines as what reduces part mixtures.

2

```
module diffmix_25px_0 (
  input wire a_fluid,
  input wire b_fluid,
  output wire out_fluid
);

  // Implementation details of the diffmix_25px_0 module

endmodule

module mix0 (
  input wire a_fluid,
  input wire b_fluid,
  output wire out_fluid
);

  diffmix_25px_0 mix0 (.a_fluid(a_fluid), .b_fluid(b_fluid), .out_fluid(out_fluid));

Endmodule

module top_module;

  wire soln1, soln2, connect0;

  mix0 mixer (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(connect0));

Endmodule
```

This one was a complete fail. It seems to be confusing the mix request. I'll do a simple reduction benchmark too. It also made its own mixer??

3

```
module chip_design (
  input wire soln1,
  input wire soln2,
  output wire connect0
);

  // Mixer component for mixing two solutions
  diffmix_25px_0 mix0 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(connect0));

endmodule
```

Fail

4

```
module top_module;

  // Define the inputs and outputs
  wire [1:0] soln1;
  wire [1:0] soln2;
  wire [1:0] connect0;

  // Instantiate the mixer component
  diffmix_25px_0 mix0 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(connect0));

Endmodule
```

Fail

5

```
module my_chip (
  input wire solnA,
  input wire solnB,
  output wire mixed_soln
);

  diffmix_25px_0 mix0 (.a_fluid(solnA), .b_fluid(solnB), .out_fluid(mixed_soln));

Endmodule
```

Fail.

Mix two parts of solution x with one part of two other mixed solutions

```
module chip_design(
  input wire soln1,
  input wire soln2,
  input wire soln3,
  output wire connect0
);

  // Mixer instances
  diffmix_25px_0 mix0 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(mix_out0));
  diffmix_25px_1 mix1 (.a_fluid(soln1), .b_fluid(soln3), .out_fluid(mix_out1));

  // Serpentine instances
  serpentine_50px_0 serp0 (.in_fluid(mix_out0), .out_fluid(serp_out0));
  serpentine_50px_1 serp1 (.in_fluid(mix_out1), .out_fluid(serp_out1));

  // Mixer instance for final mixing
  diffmix_25px_2 mix2 (.a_fluid(serp_out0), .b_fluid(serp_out1), .out_fluid(connect0));

Endmodule
```

This is actually technically a win, although it does not correctly use components as specified, and has an initialization error

2

```
module chip_design(
  input wire soln1,
  input wire soln2,
  output wire connect0
);

  // Mixer components
  diffmix_25px_0 mix0 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(connect0));

Endmodule
```

Flat fail

3

```
module chip_design (
  input wire [7:0] soln1,
  input wire [7:0] soln2,
  output wire [7:0] output
);

  // Mixer components
  diffmix_25px_0 mix0 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(output));
  diffmix_25px_1 mix1 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(output));
  diffmix_25px_2 mix2 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(output));

  // Serpentine components
  serpentine_50px_0 serp0 (.in_fluid(output), .out_fluid(output));
  serpentine_50px_1 serp1 (.in_fluid(output), .out_fluid(output));

Endmodule
```

This one totally sucked. Also, the components are improperly labeled.

4

```
module chip_design(
  input wire soln1,
  input wire soln2,
  output wire connect0
);

  // Mixer 1
  diffmix_25px_0 mix0 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(temp0));

  // Mixer 2
  diffmix_25px_1 mix1 (.a_fluid(temp0), .b_fluid(temp0), .out_fluid(connect0));

Endmodule
```

Fail.

5

```
module chip_design (
  input wire soln1,
  input wire soln2,
  output wire connect0
);

  // Mixer components
  diffmix_25px_0 mix0 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(connect0));

Endmodule
```

Fail
There was one really cool success and 4 flat fails

reduce solution A by 2 and then mix it with solution B

```
// Define the components
diffmix_25px_0 mix0 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(connect0));
serpentine_50px_0 serp1 (.in_fluid(soln1), .out_fluid(reduced_soln1));
diffmix_25px_0 mix1 (.a_fluid(reduced_soln1), .b_fluid(soln2), .out_fluid(final_soln));

// Connections
assign soln1 = A;
assign soln2 = B;
```

This is a fail because it doesn't follow complete verilog format (disobeys prompt). It is close on reduction, were there to be no serpentine this would be technically correct, with one part solution a and two parts solution b.

2

```
diffmix_25px_0 mix0 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(connect0));
serpentine_50px_0 serp1 (.in_fluid(soln1), .out_fluid(soln1_diluted));
serpentine_50px_0 serp2 (.in_fluid(soln1_diluted), .out_fluid(soln1_reduced));
diffmix_25px_0 mix1 (.a_fluid(soln1_reduced), .b_fluid(soln2), .out_fluid(output))
```

Same fail on structure and verilog

3

```
module lab_chip(
  input [7:0] soln1,
  input [7:0] soln2,
  output [7:0] connect0
);

  // Serpentine to reduce solution A by 2
  serpentine_50px_0 serp1(.in_fluid(soln1), .out_fluid(diluted_soln1));

  // Mixer to mix diluted solution A with solution B
  diffmix_25px_0 mix0(.a_fluid(diluted_soln1), .b_fluid(soln2), .out_fluid(connect0));

Endmodule
```

Fail on account of multiple bit use, PASS ON STRUCTURE!

4

```
// Serpentine to dilute solution A by half
serpentine_50px_0 serpA(.in_fluid(solnA), .out_fluid(dilutedA));

// Mixer to mix diluted solution A with solution B
diffmix_25px_0 mixAB(.a_fluid(dilutedA), .b_fluid(solnB), .out_fluid(mixedAB));
```

Fail on account of incomplete verilog but also PASS ON STRUCTURE

5

```
// Component Declarations
diffmix_25px_0 mix0 (.a_fluid(soln1), .b_fluid(soln2), .out_fluid(connect0));
serpentine_50px_0 serp1 (.in_fluid(soln1), .out_fluid(reduced_soln1));
diffmix_25px_0 mix1 (.a_fluid(reduced_soln1), .b_fluid(soln2), .out_fluid(final_mix));

// Connections
```

Same fails as earlier

Interesting takeaways:

Sometimes the LLM intuited non specified solutions to the problem. This was very impressive. How do we get our success rate up is the question for future work.