NYU

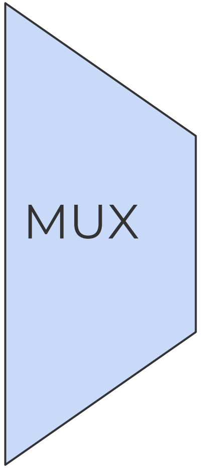
Cognichip Hackathon: Use of ROME automated hierarchical prompting technique to generate high-complexity cryptography core

Andre Nakkab

Existing SotA: Flat Prompting

Target Module

64-to-1 Mux



Module Prompt

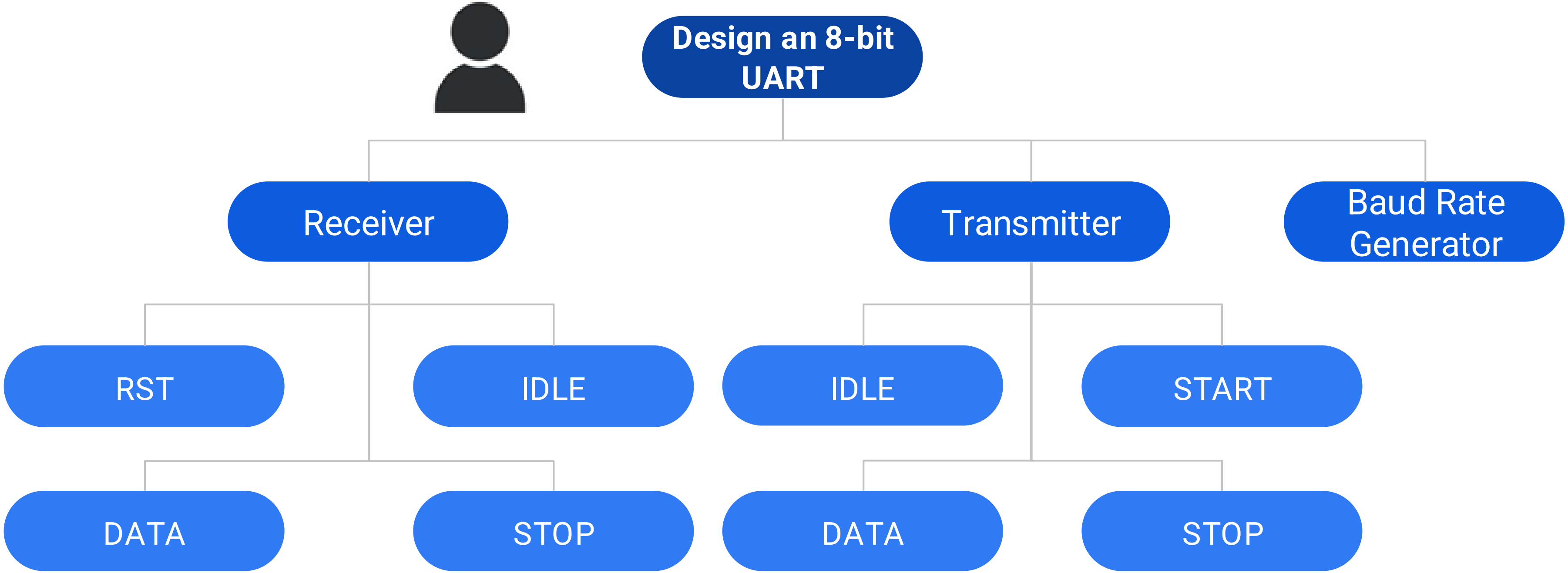
We will be generating a 64-to-1 multiplexer in Verilog. The module is defined as:

```
module mux64_1(
input [63:0] data_in,
input [5:0] select,
output reg data_out
)
```

GPT-4 Output

```
module mux64to1(input [63:0] in, input
[5:0] select, output reg out);
always @ (in or select)
begin
   case(select)
      6'd0 : out = in[0];
      6'd1 : out = in[1];
      6'd2 : out = in[2];
      6'd3 : out = in[3];
      // Continue specifying each input...
      6'd60: out = in[60];
      6'd61: out = in[61];
      6'd62: out = in[62];
      6'd63: out = in[63];
      default: out = 1'bx;     endcase
end
endmodule
```

But How Do Humans Code?



Source: https://github.com/hell03end/verilog-uart

Our Solution: Hierarchical Prompting

Global Prompt: We will be designing a 64-to-1 multiplexer in Verilog using hierarchical submodules.

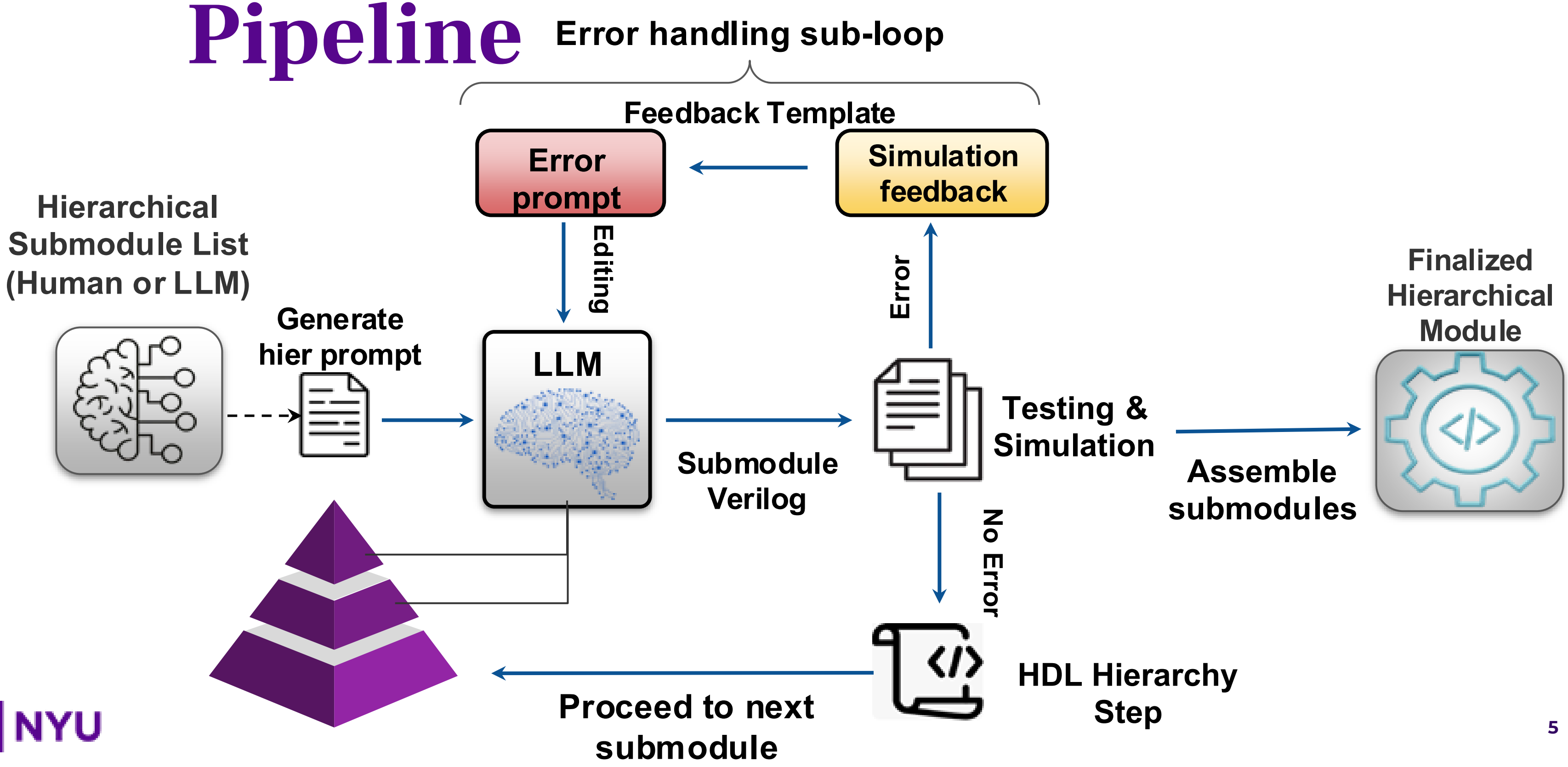
We begin by generating a 2-to-1 multiplexer with the following structure…

We have access to a 2-to-1 multiplexer, and a 4-to-1 multiplexer defined as…

Using these modules, we generate an 8-to-1 multiplexer…

```
module Mux64to1(
input [63:0] in, input [5:0]
sel, output out);
   wire out0, out1;
   Mux32to1 MUX0 ( …
```

ROME Hierarchical Pipeline



Error handling sub-loop

Feedback Template

Error prompt

Simulation feedback

Editing

Error

Hierarchical Submodule List (Human or LLM)

Generate hier prompt

LLM

Submodule Verilog

Testing & Simulation

Finalized Hierarchical Module

Assemble submodules

No Error

HDL Hierarchy Step

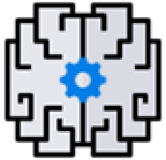
Proceed to next submodule

ROME Modes of Operation

Human-driven Hierarchical Prompting (HDHP)

Human provides hierarchy ✗
They can also provide testbenches for unit tests, improving performance ✓

Purely-generative Hierarchical Prompting (PGHP)



LLMs generates hierarchy! ✓
Only top-level testbench provided by human but unit tests must be LLM-generated ✗

Initial comparison:

Pass rate of platform	ROME w/ iVerilog & GPT-5.2	ROME w/ Cognichip
64-to-1 Mux	1.0	1.0
5-to-32 decoder	1.0	1.0
32-bit barrel shifter	1.0	1.0
128-bit AES block cipher	0.7	0.8

Novel LLM-generated Hardware

Pass rate of platform	ROME w/ iVerilog & GPT-5.2	ROME w/ Cognichip
Combined 128- and 256-bit AES block cipher	0.4	(one partial completion before encountering daily message limit on last day)

Cognichip implementation successfully produces encipher block ✓

Pros & Cons:

- Inconsistency with simulation server (common connectivity & overuse issues), but built in simulator that LLM can consistently execute is great!

- Daily message limits (makes automated error handling less of an option), but high-performance reasoning LLM-coding assistant built into IDE is great!

Conclusions:

- Cognichip is a super convenient platform (pro version would be better)

- Competitive with/exceeds open-source & some commercially available options

- Hierarchical prompting serves as a great force-multiplier for the platform

- An API for use in scripting would be a major tool

GitHub Repo:

https://github.com/ajn313/cognichip-hackathon