

# MetRex: A Benchmark for Verilog Code Metric Reasoning using LLMs

---

Manar Abdelatty, Jingxiao Ma, Sherief Reda

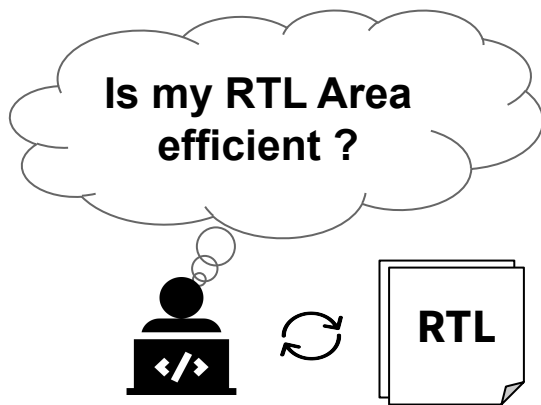
Brown University, Providence, RI



BROWN

# Early PPA Estimation Of Verilog Designs

- **Motivation:** Provide designer with early feedback on the quality (power, performance, area) of their designs by avoiding expensive synthesis time.



⚡ Faster Design Cycles

## RTL Style 3

```
1 module CarryLookAheadAdder (  
2   input [7:0] A, input [7:0] B, input cin  
3   output [7:0] Sum, output cout  
4 );  
5   for (i = 0; i < 8; i = i + 1) begin  
6     assign g[i] = a[i] & b[i];  
7     assign p[i] = a[i] ^ b[i];  
8   end  
9   assign c[1] = g[0] | (p[0] & c[0]);  
10  .....  
11  assign c[1] = g[0] | (p[0] & c[0]);  
12  assign s = a ^ b ^ c[7:0];  
13 endmodule  
14  
15 endmodule
```

⚡ Architectural Choice Tradeoffs

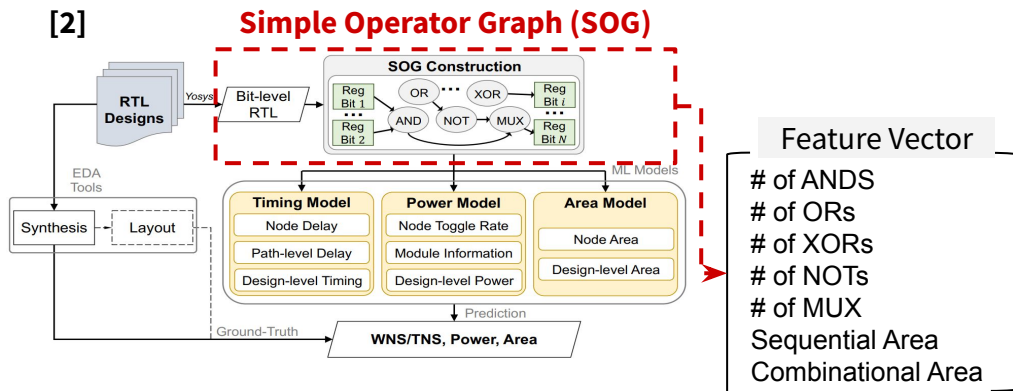
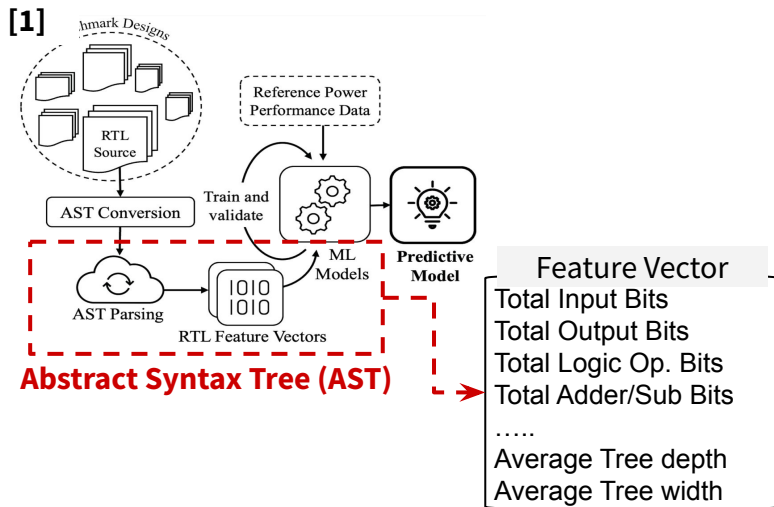
## RTL Style 4

```
1 module CarryLookAheadAdder (  
2   input [7:0] A, input [7:0] B, input cin  
3   output [7:0] Sum, output cout  
4 );  
5   for (i = 0; i < 8; i = i + 1) begin  
6     assign g[i] = a[i] & b[i];  
7     assign p[i] = a[i] ^ b[i];  
8   end  
9   assign c[1] = g[0] | (p[0] & c[0]);  
10  .....  
11  assign c[1] = g[0] | (p[0] & c[0]);  
12  assign s = a ^ b ^ c[7:0];  
13 endmodule  
14  
15 endmodule  
16  
17 assign cout = carry[8];  
18 endmodule
```

# Previous Work: Metric Estimation Using Machine Learning

~~Input: RTL Code~~  $\xrightarrow{\text{Predict}}$  ~~Post synthesis Metrics (Area, Delay, Power)~~

RTL Code  $\xrightarrow{\text{Preprocess}}$  **Input: Feature Vector**  $\xrightarrow{\text{Predict}}$  **Output: Post-synthesis Metrics**



[1] P. Sengupta, et al. "How Good Is Your Verilog RTL Code? A Quick Answer from Machine Learning," 2022 IEEE/ACM International Conference On Computer Aided Design (ICCAD), San Diego, CA, USA.

[2] Fang, Wenji, et al. "MasterRTL: A Pre-Synthesis PPA Estimation Framework for Any RTL Design." 2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD). IEEE, 2023.

# Previous Work: Metric Estimation Using Machine Learning

- **Intermediate Formats:** Have to convert RTL code to intermediary format like Abstract Syntax Tree (ASTs) or Simple Operator Graphs (SoG).
- **Manual Feature Extraction:** Extract manually engineered features from the intermediate format; extracted features constitute the input to the ML model.

# What LLMs Could Offer ?

⚡ Process RTL code directly (a **lossless** representation):

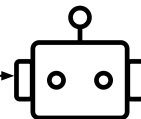
- Eliminate the need for manual feature extraction and conversion into intermediary format.
- LLM can autonomously extract relevant patterns from RTL code.
- Provide a potentially faster and insightful analysis.

## Input

```
1 module CarryLookAheadAdder (  
2   input [7:0] A, input [7:0] B, input cin  
3   output [7:0] Sum, output cout  
4 );  
5  
6 for (i = 0; i < 8; i = i + 1) begin  
7   assign g[i] = a[i] & b[i];  
8   assign p[i] = a[i] ^ b[i];  
9 end  
10 assign c[1] = g[0] | (p[0] & c[0]);  
11 .....  
12 assign c[1] = g[0] | (p[0] & c[0]);  
13 assign s = a ^ b ^ c[7:0];  
14 endmodule
```

//

**No Preprocessing**  
**No Need for intermediary formats**



After synthesis the  
design will have..  
Thus total are will  
be 12.0



# Introducing MetRex

- LLM-based Verilog code metric reasoning

**How effectively can Large Language Models (LLMs) reason about post-synthesis PPA metrics of Verilog Designs ?**

# Dataset Collection & Cleaning

- Syntax errors and Synthesis warning were automatically fixed by LLM.
- All 25,868 Designs are syntax-error free & *synthesizable*.

**Table 1.** HDL design sources.

Source	Designs <sup>1</sup> (Count)	Complexity (Code Length) {Min, Median, Max}
RTL-Coder <sup>2</sup> [21]	18,450	{3, 29, 918}
VeriGen [5]	7,292	{5, 69, 27,025}
ISCAS'89 [22]	29	{53, 530, 54,778}
ISCAS'85 [22]	10	{17, 1225, 3925}
OpenCores [23]	54	{1, 103, 2716}
NVLA [24]	33	{19, 1333, 42,051}
<b>MetRex (ours)</b>	25,868 (Train) 138* (Test)	{3, 35, 54,778}

**25,868 Designs**  
**Syntax-Error Free & Synthesizable**



**NVLA.org**

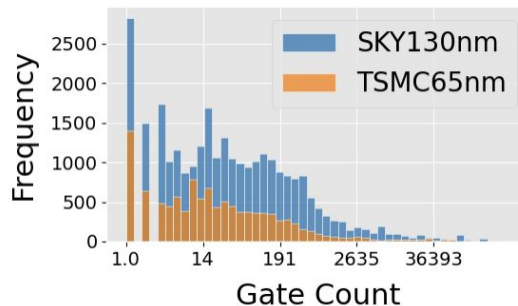
[1] Liu, Shang, et al. "Rtlcoder: Outperforming gpt-3.5 in design rtl generation with our open-source dataset and lightweight solution." 2024 IEEE LLM Aided Design Workshop (LAD). IEEE, 2024.

[2] Thakur, Shailja, et al. "Verigen: A large language model for verilog code generation." ACM Transactions on Design Automation of Electronic Systems 29.3 (2024): 1-31.

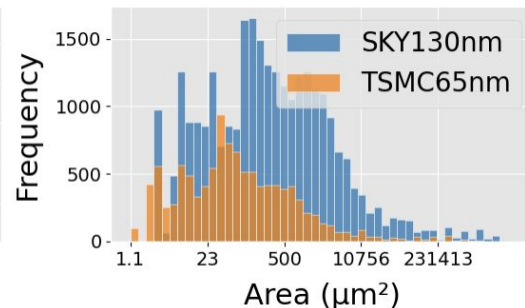
# MetRex Dataset

- **25,868** Verilog Designs
- Synthesized using *Skywater 130nm* and *TSMC 65nm*
- Annotated with their post-synthesis metrics on:
  - Area
  - Delay
  - Static Power

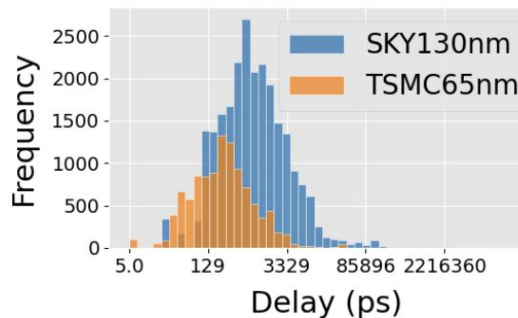
**Gate Count**



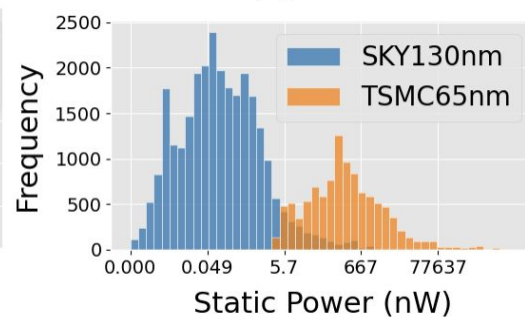
**Area**



**Delay**



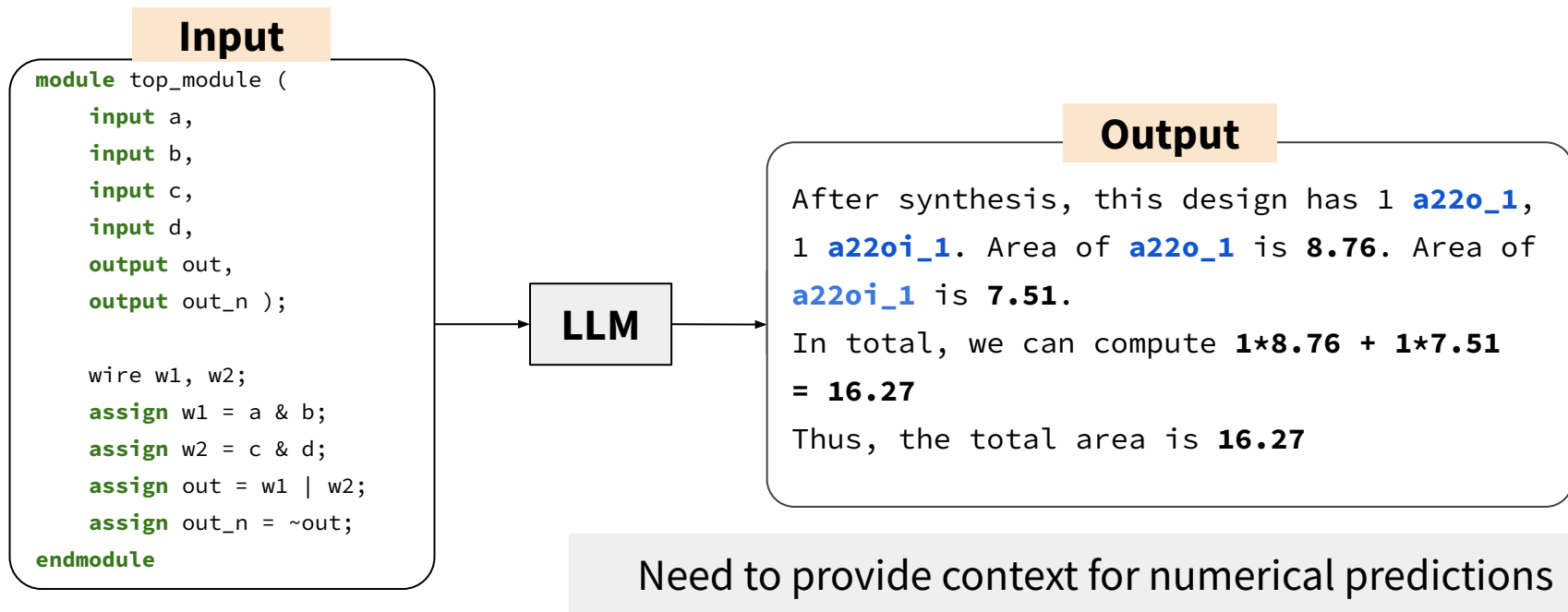
**Static Power**





# Chain Of Thoughts (CoT) Prompting

⚡ Bridge the gap between the final metrics and input RTL code



# MetRex Dataset

- **25,868** Verilog Designs
- Annotated with their post-synthesis metrics on:
  - Area
  - Delay
  - Static Power

## RTL

```
1 module adder (  
2     input a, b, cin,  
3     output cout, sum  
4 );  
5     assign sum = a ^ b ^ cin;  
6     assign cout = a & b | a & cin | b & cin;  
7 endmodule
```

## Synthesis Thought

After synthesis, this design has **1 maj3\_1**, **1 xor3\_1**.

## Area Thought

Area of **maj3\_1** is **10.01**. Area of **xor3\_1** is **23.77**. In total, we can compute **1\*10.01 + 1\*23.77 = 33.78**. Thus, total area is **33.78**.

## Delay Thought

The critical path goes through the following gates: **xor3\_1** with a fanout of **2** and a load capacitance of **0.01**. Delay of **xor3\_1** is **0.34**. In total, we can compute **0.34 = 0.34**. Thus, total delay is **0.34**.

## Static Power Thought

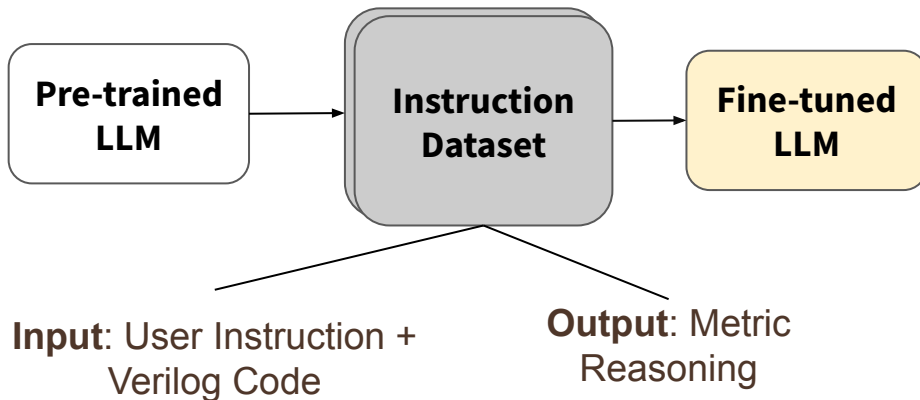
Leakage power of **maj3\_1** is **2.09**. Leakage power of **xor3\_1** is **15.22**. In total, we can compute **1\*2.09 + 1\*15.22 = 17.31**. Thus, the total static power is **17.31**.



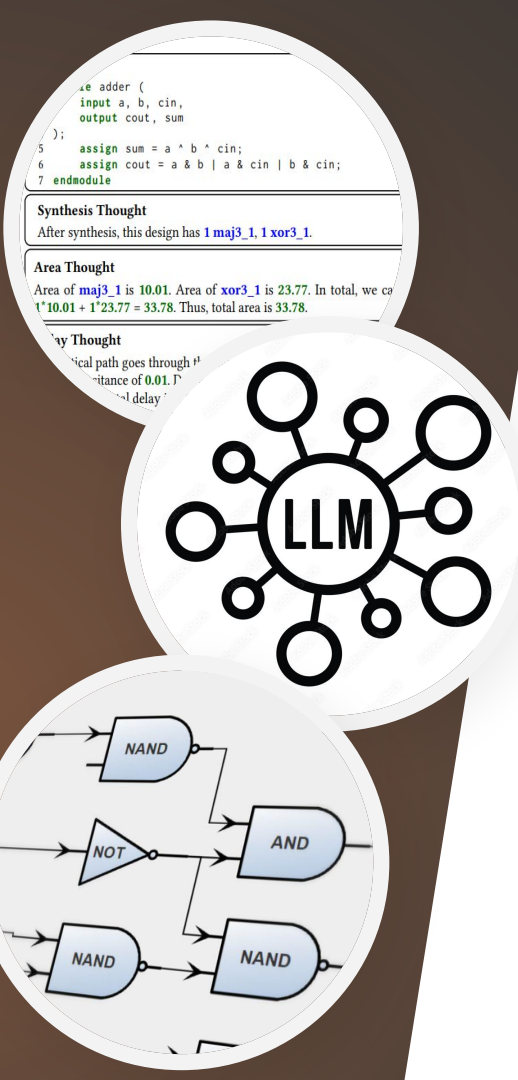
<https://huggingface.co/datasets/scale-lab/MetRex>

# Experimental Results

- Instruction Tuning



Mixtral-8x7b-Instruct  
Meta-Llama3-Instruct-8b



# Experimental Setup: Evaluation Set

- **Evaluation Set:** Derived from the VerilogEval benchmark\*
- Synthesis Tool:
  - Yosys
  - Sky130nm Tech.

**Table 2.** Test dataset derived from the VerilogEval benchmark [3], categorized by difficulty level.

Difficulty Description		#	Gate Count {Min, Med, Max}
<b>Level-1 (L1)</b>	Basic logic gates	10	{1, 1, 1}
	Multi-bit gates	9	{2, 2, 100}
	1-bit comb. circuits	5	{2, 2, 4}
	<b>Total #</b>	<b>23</b>	<b>{1, 2, 100}</b>
<b>Level-2 (L2)</b>	Adder circuits	4	{2, 6, 15}
	Multi-bit comb. circuits	23	{1, 3, 11}
	Flip-Flop registers	14	{1, 3, 24}
	Basic Seq. circuits	2	{8, 8, 8}
	<b>Total #</b>	<b>43</b>	<b>{1, 3, 24}</b>
<b>Level-3 (L3)</b>	Finite state machines	24	{3, 11, 57}
	Counters	9	{10, 14, 48}
	Complex comb. logic	29	{1, 7, 580}
	Advanced Seq. circuits	9	{11, 67, 607}
	<b>Total #</b>	<b>72</b>	<b>{1, 14, 607}</b>

\*Liu, Mingjie, et al. "VerilogEval: Evaluating large language models for verilog code generation." *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 2023.

# Experimental Setup: acc@k

Aggregate Top-K answers by taking the median of the final answer.

## User Instruction:

Estimate Area for the given verilog module:

```
module top_module(  
  input a,  
  input b,  
  output out_nand,  
  output out_nor  
);  
  assign out_nand = ~(a & b);  
  assign out_nor = ~(a | b);  
endmodule
```

LLM

After synthesis, this design has 1 **and2\_0**, 1 **nor2\_1**. Area of **and2\_0** is 6.26. Area of **nor2\_1** is 3.75. In total, we can compute  $1*6.26 + 1*3.75 = 10.01$ . Thus, the total area is **10.01**

After synthesis, this design has 1 **nand2\_1**, 1 **nor2\_1**. Area of **nand2\_1** is 3.75. Area of **nor2\_1** is 3.75. In total, we can compute  $1*3.75 + 1*3.75 = 7.5$ . Thus, the total area is **7.5**

....  
....

After synthesis, this design has 1 **and2\_0**, 1 **nor2\_1**. Area of **and2\_0** is 6.26. Area of **nor2\_1** is 3.75. In total, we can compute  $1*6.26 + 1*3.75 = 10.01$ . Thus, the total area is **10.01**

Final Answer

**10.01**

$$acc@k(MRE \leq t) = E_N \left[ \mathbb{1} \left( \frac{|median(\hat{R}_{i,1:k}) - R_i|}{R_i} \leq t \right) \right]$$

# Results

✗ Not fine tuned  
✓ Finetuned with CoT

Percentage of designs with MRE  
less-than threshold value  $t$

Margin ( $t$ ) MRE $\leq t$	Model	Finetuned ?	Area (acc@k) $\uparrow$			Delay (acc@k) $\uparrow$			Static Power (acc@k) $\uparrow$		
			acc@1	acc@5	acc@10	acc@1	acc@5	acc@10	acc@1	acc@5	acc@10
10%	Mixtral-MetRex-8x7b	✗	19.6%	19.6%	21.0%	23.9%	22.5%	22.5%	20.3%	18.8%	19.6%
		✓	43.5%	46.4%	45.7%	42.8%	45.7%	45.7%	39.1%	39.9%	38.4%
		$\Delta$	+23.9%	+26.8%	+24.6%	+18.8%	+23.2%	+23.2%	+18.8%	+21.0%	+18.8%
	LLama3-MetRex-8b	✗	17.4%	18.8%	18.1%	20.3%	23.9%	22.5%	15.9%	15.2%	15.2%
		✓	<b>58.0%</b>	<b>58.0%</b>	<b>58.7%</b>	<b>47.8%</b>	<b>47.1%</b>	<b>47.8%</b>	<b>42.0%</b>	<b>42.8%</b>	<b>41.3%</b>
		$\Delta$	+40.6%	+39.1%	+40.6%	+27.5%	+23.2%	+25.4%	+26.1%	+27.5%	+26.1%
20%	Mixtral-MetRex-8x7b	✗	25.4%	26.1%	26.1%	31.9%	29.0%	29.7%	25.4%	26.1%	28.3%
		✓	58.0%	61.6%	60.9%	50.7%	53.6%	55.8%	<b>53.6%</b>	<b>54.3%</b>	<b>52.2%</b>
		$\Delta$	+32.6%	+35.5%	+34.8%	+18.8%	+24.6%	+26.1%	+28.3%	+28.3%	+23.9%
	LLama3-MetRex-8b	✗	22.5%	23.9%	22.5%	25.4%	28.3%	28.3%	22.5%	21.7%	21.0%
		✓	<b>73.2%</b>	<b>76.1%</b>	<b>74.6%</b>	<b>61.6%</b>	<b>64.5%</b>	<b>63.8%</b>	52.2%	49.3%	47.1%
		$\Delta$	+50.7%	+52.2%	+52.2%	+36.2%	+36.2%	+35.5%	+29.7%	+27.5%	+26.1%

MRE error less than  $t$

- Finetuning improves results **37.0%**, **25.3%**, **25.7%** on area, delay, static power respectively.
- Best performing models achieves **73.2%**, **61.6%**, and **52.2%** on area, delay, static power respectively within MRE threshold of **20%**.

# Architectural Choices Evaluation

## User Instruction

Estimate area for the given RTL design. Reason about the number and type of gates that would be present after synthesis.

```
1 module BasicAdder (  
2     input [7:0] A, input [7:0] B,  
3     output [7:0] Sum  
4 );  
5     assign Sum = A + B;  
15 endmodule
```

## LLM Answer

After synthesis, this design has 6 maj2\_1, 1 o21ai\_0, 13 xnor2\_1, 2 xnor2\_0, 10.01. Area of nand2\_1 is 3.75... compute  $6 \times 10.01 + 2 \times 3.75 + \dots = 26$

Thus, the total area is **207.71**

## User Instruction

Estimate area for the given RTL design. Reason about the number and type of gates that would be present after synthesis.

```
1 module CarryLookAheadAdder (  
2     input [7:0] A, input [7:0] B, input cin  
3     output [7:0] Sum, output cout  
4 );  
5  
6     for (i = 0; i < 8; i = i + 1) begin  
7         assign g[i] = a[i] & b[i];  
8         assign p[i] = a[i] ^ b[i];  
9     end  
10    assign c[1] = g[0] | (p[0] & c[0]);  
11    .....  
12    assign c[1] = g[0] | (p[0] & c[0]);  
13    assign s = a ^ b ^ c[7:0];  
14 endmodule
```

## LLM Answer

After synthesis, this design has 8 maj3\_1, 8 xor3\_1. Area of maj3\_1 is 10.01. Area of xor3\_1 is 23.77. In total, we can compute  $8 \times 10.01 + \dots = 230.24$  Thus, the total area is 230.24

Thus, the total area is **230.24**

Estimate area for the given RTL design. Reason about the number and type of gates that would be present after synthesis.

```
[7:0] B, input cin  
output cout  
  
adder (...);  
adder_c0 (...);  
adder_c1 (...);  
  
sum_high_c1 : sum_high_c0;  
cout_high_c1 : cout_high_c0;
```

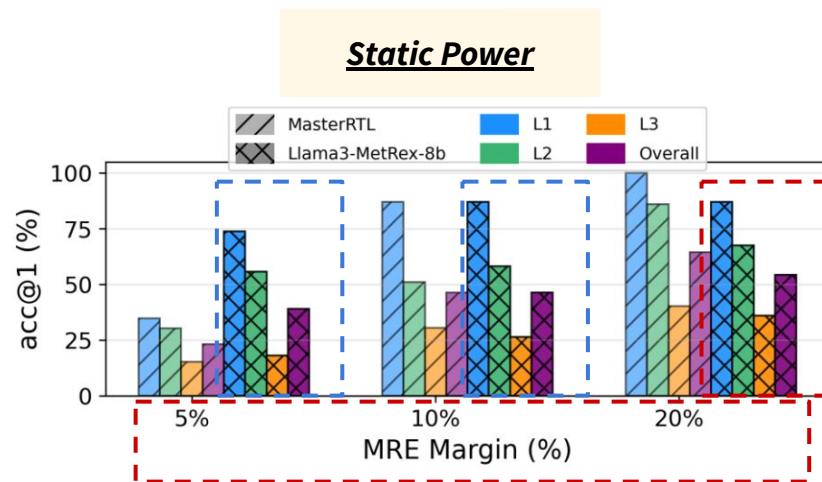
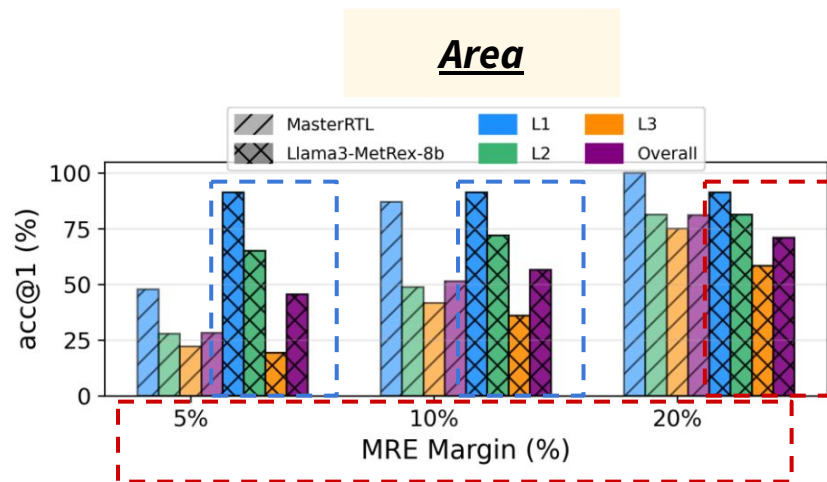
After synthesis, this design has 1 a2loi\_1, 1 a22loi\_1, 1 a2\_0, 2 clkinv\_1, 1 maj3\_1, 9 maj2\_1, 1 o21ai\_0, 1 o311a\_1, 1 o31a\_1, 1 xor2\_1. Area of a2loi\_1 is 5.0. .... In total, we can compute

**66.48**

**Ground Truth: 265.23**

**Ground Truth: 220.24**

# Comparison to Regression Based Models: acc@1

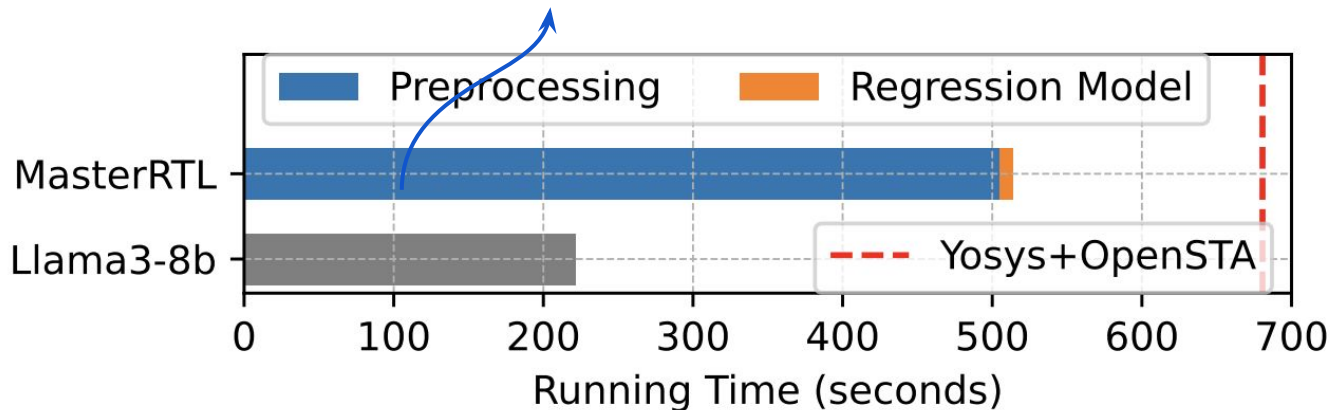


- ➔ LLM outperform under tight error margins 5% and 10% error margins.
- ➔ Regression models perform better under more relaxed error constraints (20% error margin)



# Comparison to Regression Based Models: Inference Runtime

## Conversion to Simple Operator Graph (SOG) and feature extraction



**2x** Faster than logic synthesis, and **1.7x** Faster than regression based models.

# Conclusion

---

- We introduced an open-source dataset **MetRex**, 25,868 synthesizable designs, annotated with their post-synthesis metrics.
- We showed that supervised finetuning can improve LLMs performance on the metric estimation task by **37.0%**, **25.3%**, **25.7%** on area, delay, static power respectively.
- Best performing models achieves **73.2%**, **61.6%**, and **52.2%** on area, delay, static power respectively within MRE threshold of **20%**.

# Thank You!



<https://github.com/scale-lab/MetRex>

