



香港中文大學

The Chinese University of Hong Kong

# Large Language Model in EDA

Bei Yu

Department of Computer Science & Engineering

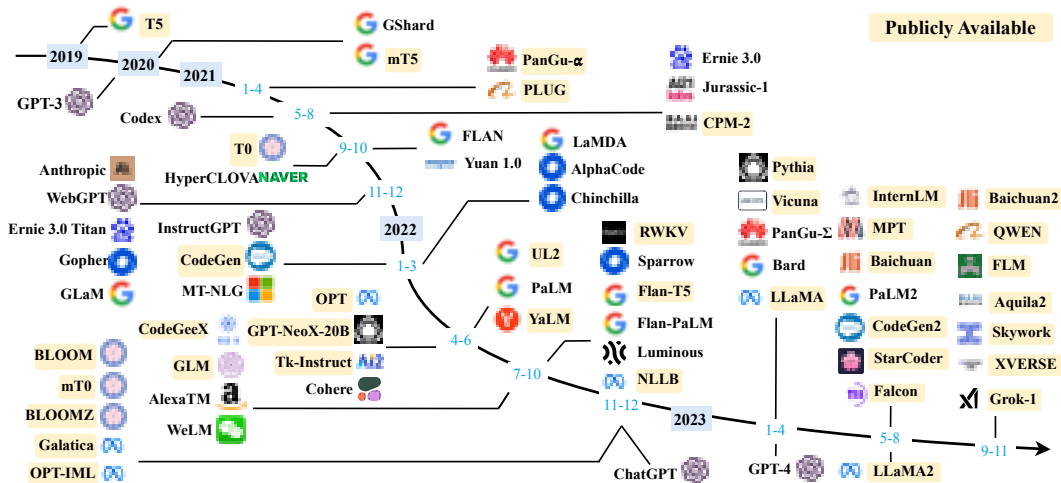
Chinese University of Hong Kong

[byu@cse.cuhk.edu.hk](mailto:byu@cse.cuhk.edu.hk)

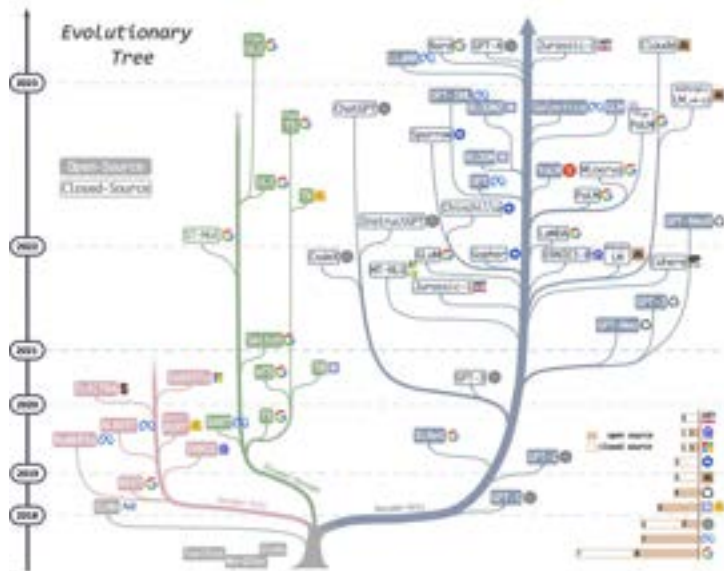


- CLI** Command Line Interface
- GUI** Graphical User Interface
- CUI** Conversational User Interface

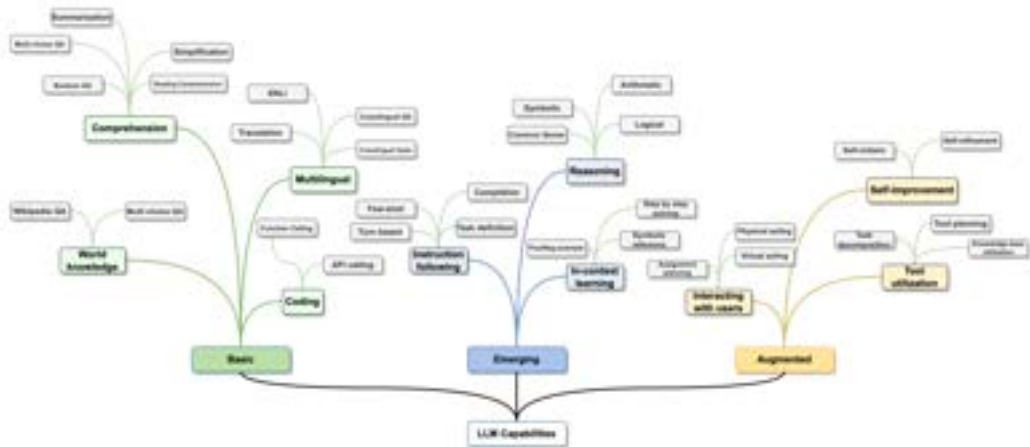
# LLM Development Timeline<sup>1</sup>



Wayne Xin Zhao et al. (2023). "A survey of large language models". In: *arXiv preprint arXiv:2303.18223*.



Jingfeng Yang et al. (2024). "Harnessing the power of LLM in practice: A survey on ChatGPT and beyond". In: *ACM TKDD* 18.6, pp. 1–32.





```
矩阵 a = tensor([ 0.3148, -0.1578, -0.2662, 0.0606, -0.4005],
 [ 1.2766, -0.0564, -2.4822, -0.5052, -0.5892],
 [-0.3722, -0.1244, -0.6636, 0.8963, -0.8977],
 [ 1.8294, -0.6872, -0.0040, -0.8030, -1.2962]) 矩阵 b = tensor([ 1.6289, -1.8879],
 [ 1.4068, 0.8827],
 [ 8.4747, -0.0830],
 [ 0.7047, 0.7294],
 [-1.6289, 0.2868]) 使用 einsum() 来计算 a@b 的矩阵乘法
```

矩阵 A 是一个 4x5 矩阵，矩阵 B 是一个 5x2 矩阵，由于矩阵 A 的列数 (5) 与矩阵 B 的行数 (5) 相等，所以这两个矩阵可以相乘。乘积矩阵的大小为 4x2。

矩阵相乘的方法是将第一个矩阵的每一行与第二个矩阵的每一列相乘，然后求和。以下是矩阵相乘的结果：

```
[ 1. 0. 1.878, -0.6618],
 [ 1. 0.421, -1.3927],
 [ 8. 3561, 8. 1356],
 [ 0. 9432, 2. 9852]
```

Correct Ans:

$$\begin{bmatrix} 1.2766, & 0.0329 \\ 6.1799, & -2.6547 \\ -0.8886, & 0.1450 \\ 0.8300, & -0.1394 \end{bmatrix}$$

# Challenges and Improvements



```
矩阵 a = torch.tensor([ 0.3344, -0.1275, -0.0902, 0.4806, -0.4005],
                        [ 1.2766, -0.0594, -2.4632, -0.0092, -1.5892],
                        [-0.3722, -0.0344, -0.6625, 0.6963, -0.8775],
                        [-1.4294, -0.6973, -0.0049, -0.8030, -1.2963]) 矩阵 b = torch.tensor([ 1.6391, -0.8751],
                        [ 1.4066, 0.8627],
                        [ 0.4447, -0.0630],
                        [ 0.7041, 0.0294],
                        [-1.6206, 0.2690]) 请计算 a 和 b 的矩阵乘法
```

```
根据您提供的矩阵 - 计算矩阵 a 乘以矩阵 b 的矩阵乘法结果:
```

```
import torch

a = torch.tensor([ 0.3344, -0.1275, -0.0902, 0.4806, -0.4005],
                  [ 1.2766, -0.0594, -2.4632, -0.0092, -1.5892],
                  [-0.3722, -0.0344, -0.6625, 0.6963, -0.8775],
                  [-1.4294, -0.6973, -0.0049, -0.8030, -1.2963])

b = torch.tensor([ 1.6391, -0.8751],
                  [ 1.4066, 0.8627],
                  [ 0.4447, -0.0630],
                  [ 0.7041, 0.0294],
                  [-1.6206, 0.2690])

result = torch.matmul(a, b)

print(result)
```

1. Regenerate response

Correct Ans:

$$\begin{bmatrix} 1.2766, & 0.0329 \\ 6.1799, & -2.6547 \\ -0.8886, & 0.1450 \\ 0.8300, & -0.1394 \end{bmatrix}$$



## R-CNN: Region-Based CNN

Input  
image



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.  
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

Slides from Justin Johnson

Machine Learning is to **Fit** A Function  $f(x)$



## R-CNN: Region-Based CNN



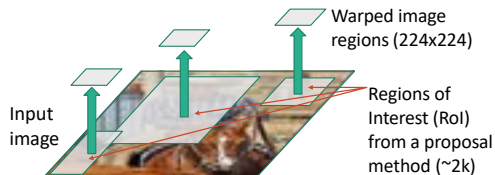
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.  
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

Slides from Justin Johnson

Machine Learning is to **Fit** A Function  $f(x)$



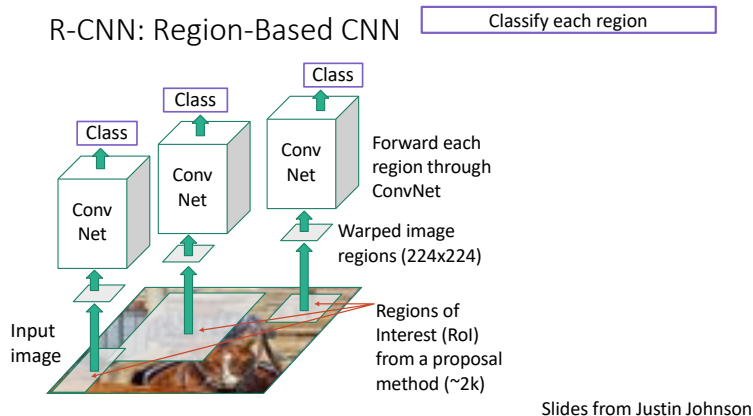
## R-CNN: Region-Based CNN



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.  
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

Slides from Justin Johnson

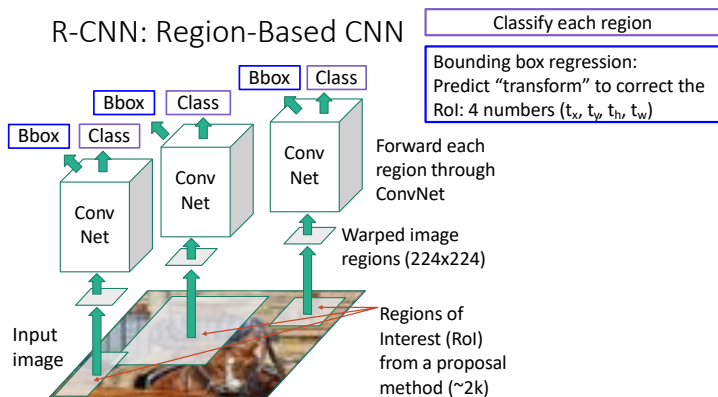
Machine Learning is to **Fit** A Function  $f(x)$



Slides from Justin Johnson

Machine Learning is to **Fit** A Function  $f(x)$

# How Machine Can Understand Picture?



Slides from Justin Johnson

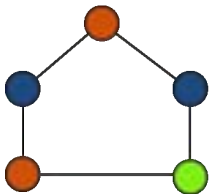
Machine Learning is to **Fit** A Function  $f(x)$

# How Machine Can Understand Text?



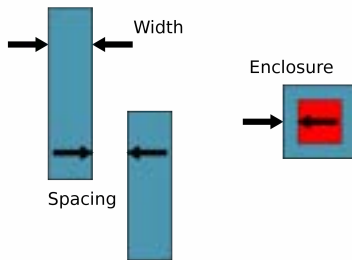
Machine Learning is to Fit A Function  $f(x)$

- Combinatorial Problem (e.g. Coloring)
- Handling Complicated Rules



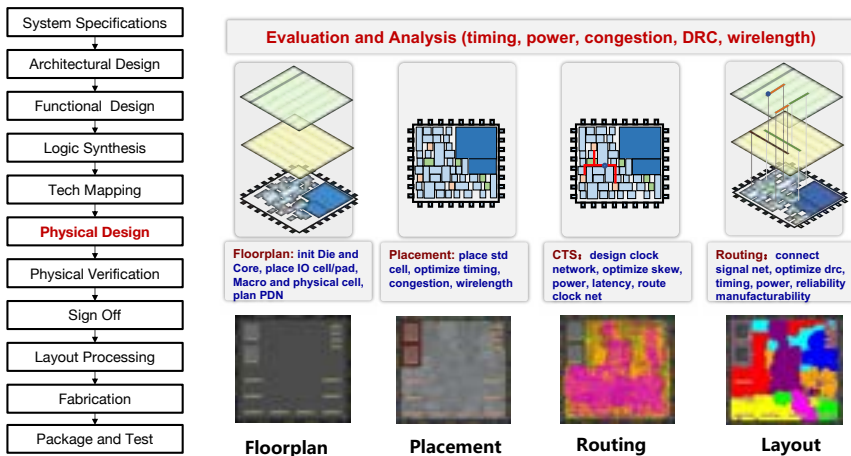
(a) Graph Coloring

## The three basic DRC checks



(b) Design Rule Checking

A typical EDA flow...<sup>4</sup>

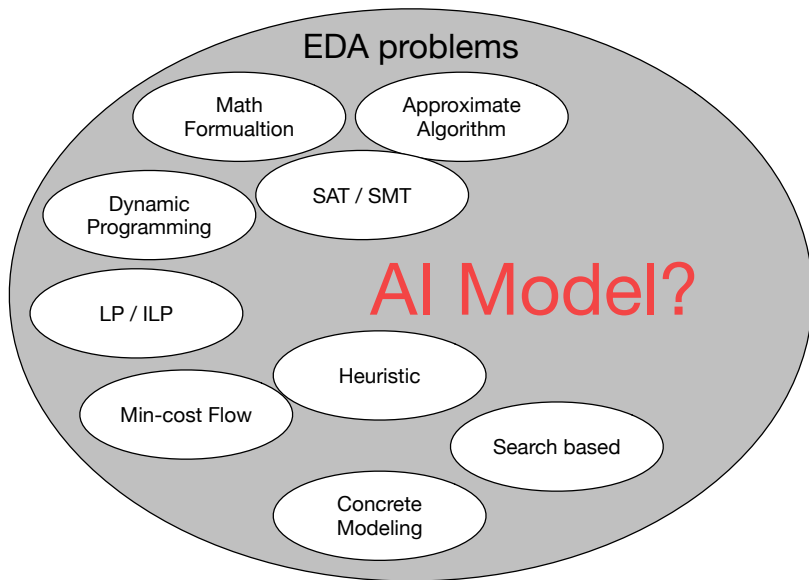


<sup>4</sup>Xingquan Li et al (2024). "iEDA: An Open-Source Intelligent Physical Implementation Toolkit and Library". In: *Proc. ASPDAC*.



EDA problems

AI Model?



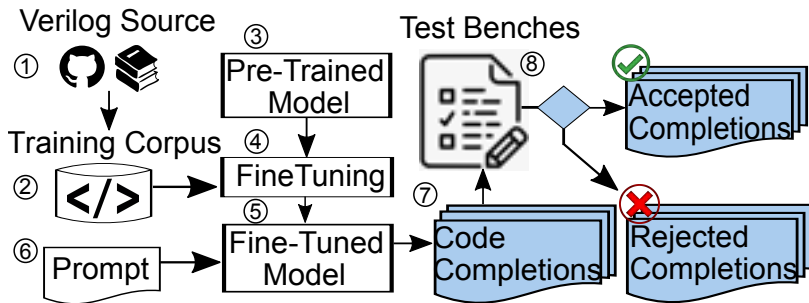


① RTL Generation and Debug

② Flow Script Generation

③ Documentation QA

# RTL Generation and Debug



Experimental Evaluation of LLM Verilog Completions.



```

module example(
  input [31:0] x,
  output [31:0] y,
  output [31:0] z,
  output [31:0] w
);
assign y = 9 * x ;
assign z = 23 * x ;
assign w = 81 * x ;
endmodule

```

Before rewriting



```

module example(
  input [31:0] x,
  output [31:0] y,
  output [31:0] z,
  output [31:0] w
);
assign y = 8 * x + x;
assign z = 32 * x - 9 * x;
assign w = 8 * (9 * x) + 9 * x;
endmodule

```

After rewriting

```

module example(
  input wire s,
  input wire [31:0] A, B, C, D,
  output reg [32:0] Z
);
always @(s, A, B, C, D) begin
  if (s)
    Z <= A + B;
  else
    Z <= C + D;
end
endmodule

```

Before rewriting



```

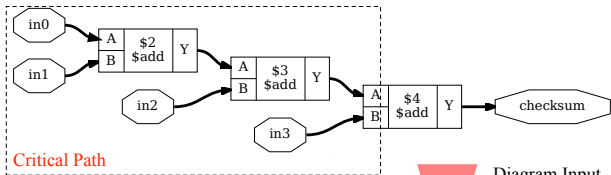
module example(
  input wire s,
  input wire [31:0] A, B, C, D,
  output reg [32:0] Z
);
reg [31:0] Z1, Z2; // Intermediate values

always @(s, A, B, C, D) begin
  if (s) begin
    Z1 <= A;
    Z2 <= B;
  end else begin
    Z1 <= C;
    Z2 <= D;
  end
end

always @(Z1, Z2) begin
  Z <= Z1 + Z2;
end
endmodule

```

After rewriting

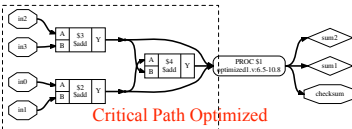
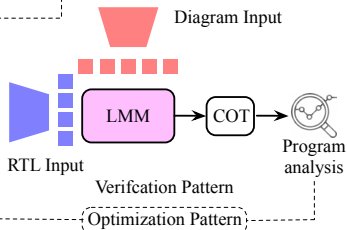


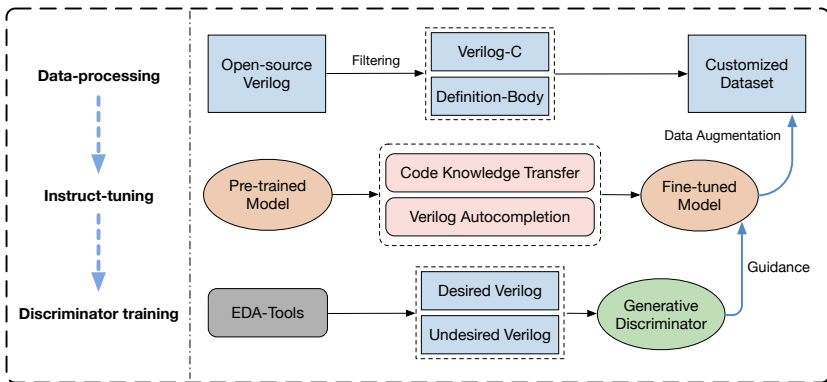
```

module checksum_basic(
    input [15:0] in0, in1, in2, in3,
    output reg [17:0] checksum
);
    always #* begin
        checksum = in0 + in1 + in2 + in3;
    end
endmodule
    
```

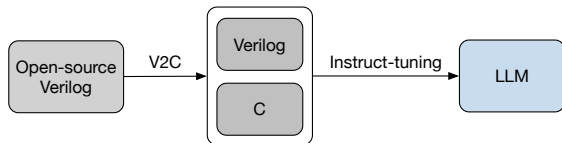
```

module checksum_basic(
    input [15:0] in0, in1, in2, in3,
    output reg [17:0] checksum
);
    reg [16:0] sum1, sum2;
    always #* begin
        sum1 = in0 + in1;
        sum2 = in2 + in3;
        checksum = sum1 + sum2;
    end
endmodule
    
```





Align Verilog with C to generate self-supervised dataset.



- Tool **V2C** to convert the Verilog into C
- Verilog-C pairs are used for LLMs **instruct-tuning**
- Alignment from C to Verilog.

## System Prompt:

You are an experienced Verilog engineer.

## Instruction:

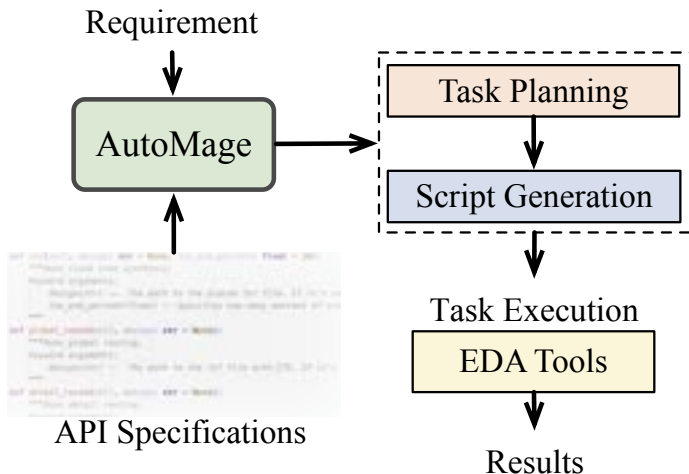
You are supposed to translate the following Verilog into C program.

```
module d_latch(  
    input d,  
    input en,  
    input rstn,  
    output reg q  
);  
  
always @ (en or rstn or d)  
    if (!rstn)  
        q <= 0;  
    else  
        if (en)  
            q <= d;  
  
endmodule
```

## Answer:

```
#include <stdio.h>  
#include <stdbool.h>  
  
void d_latch(  
    int d,  
    bool en,  
    bool rstn,  
    int *q) {  
  
    if (!rstn) {  
        *q = 0;  
    } else if (en) {  
        *q = d;  
    }  
}
```

# Flow Script Generation





## #1: User Requirement

For the design named "aes" on the platform "asap7", please perform synthesis with a clock period of 5, followed by floorplan with a core utilization of 70%. Then, execute placement with a density of 0.8. Next, proceed with CTS to fix 40% of violating paths. Finally, evaluate the performance after routing using "power" metric.

## #2: Task Decomposition

task1: set up the EDA tool  
func: set\_up()  
args:  
  design\_name: "aes"  
  platform: "asap7"

task2: perform synthesis  
func: run\_synthesis()  
args:  
  clock\_period: 5

task3: execute floorplan  
func: floorplan()  
args:  
  core\_utilization: 70

task4: perform placement  
func: placement()  
args:  
  density: 0.8

task5: perform CTS  
func: cts()  
args:  
  tns\_end\_percent: 40

task6: perform routing  
func: global\_route()  
  detail\_route()

task7: evaluation  
func: get\_metric()  
args:  
  stage: "route"  
  metrics: ["power"]

## #3: Script Generation

```
# Initialize
eda = chatEDA()

# Set up the EDA tool
eda.setup(design_name="aes", platform="asap7")

# Perform synthesis
eda.run_synthesis(clock_period=5)

# Execute floorplan
eda.floorplan(core_utilization=70)

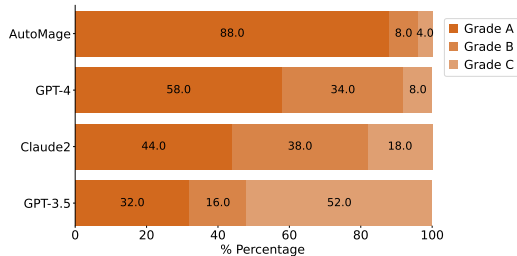
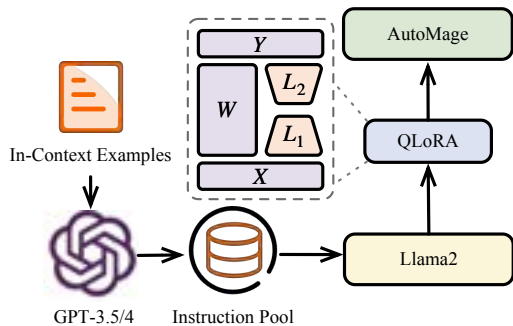
# Perform placement
eda.placement(density=0.8)

# Perform CTS
eda.cts(tns_end_percent=40)

# Perform routing
eda.global_route()
eda.detail_route()

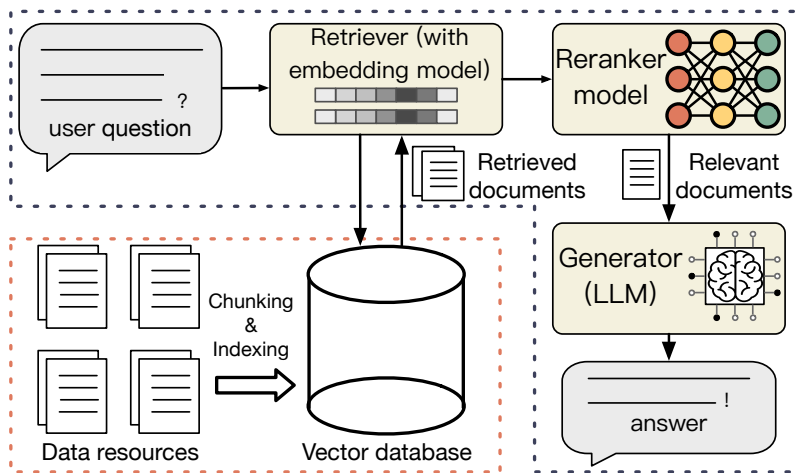
# Evaluate the performance after routing
Performance = eda.get_metric("route", ["power"])
```

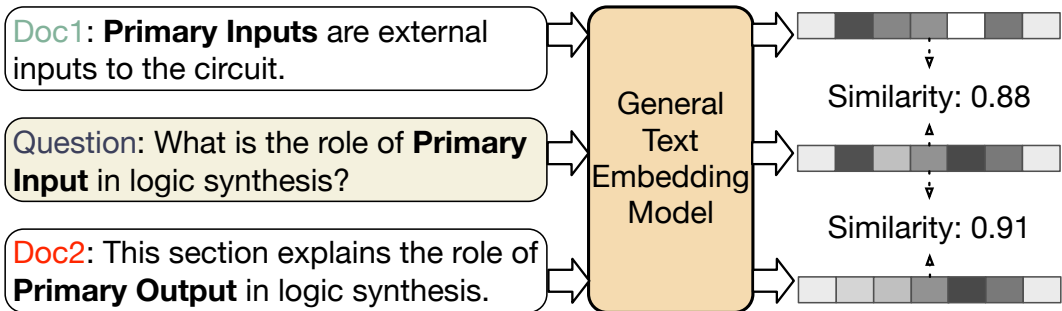




- **Grade A:** correct in planning and script
- **Grade B:** correct **only** in planning
- **Grade C:** fault in both

# Documentation QA





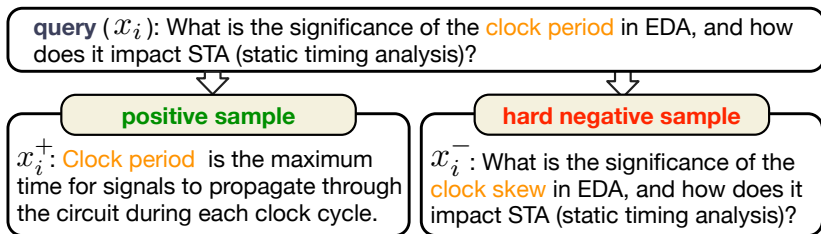


Rationale: improve semantic understanding for domain terminologies and concepts

- $x_i$ : EDA-related query
- $x_i^+$  (pos sample): answer to  $x_i$
- $x_i^-$  (neg sample): Query with another EDA term

- Loss function (in-batch negative+hard negative):

$$-\log \frac{e^{\text{sim}(x_i, x_i^+)/\tau}}{\sum_{j=1}^M (e^{\text{sim}(x_i, x_j^+)/\tau} + e^{\text{sim}(x_i, x_j^-)/\tau})}$$



# Conclusion

- RTL Automatic Generation
- EDA Tool Script Generation
- RTL&EDA Flow Debug

## Other applications:

- Design and Tool Parameter Optimization
- Layout-level Optimization
- **Education:** toward Customers or Universities



- Understand and customize RTL
- Understand and customize complicated EDA flow
- **Security**: How not to hallucinate
- **Privacy**: How not to leak IP
- Ethical use of AI

