RTLMarker: Protecting LLM-Generated RTL Copyright via a Hardware Watermarking Framework

Kun Wang, Kaiyan Chang, Mengdi Wang, Xingqi Zou, Haobo Xu, Yinhe Han, **Ying Wang***

中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

中国科学院大学
University of Chinese Academy of Sciences

智能计算机研究中心
Research Center for Intelligent Computing Systems

Outline

中国科学院大学
University of Chinese Academy of Sciences

① Introduction

② Background

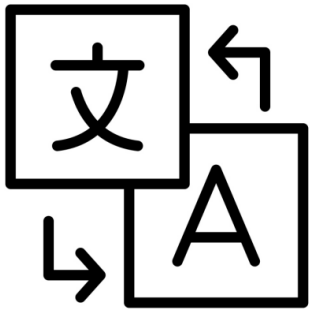
③ RTLMarker

④ Evaluation

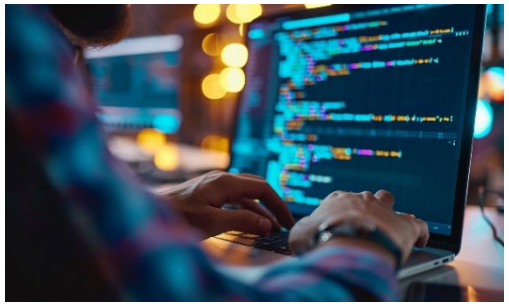
⑤ Conclusion

Large Language Model

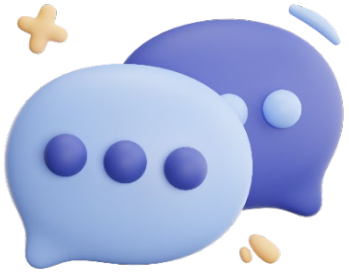
中国科学院大学
University of Chinese Academy of Sciences



Translation



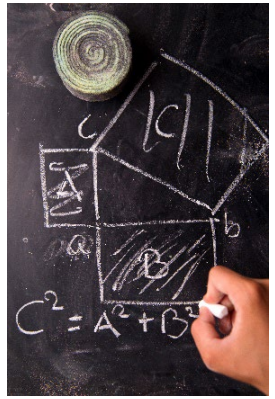
Code Generation



Chat

Summary

Search

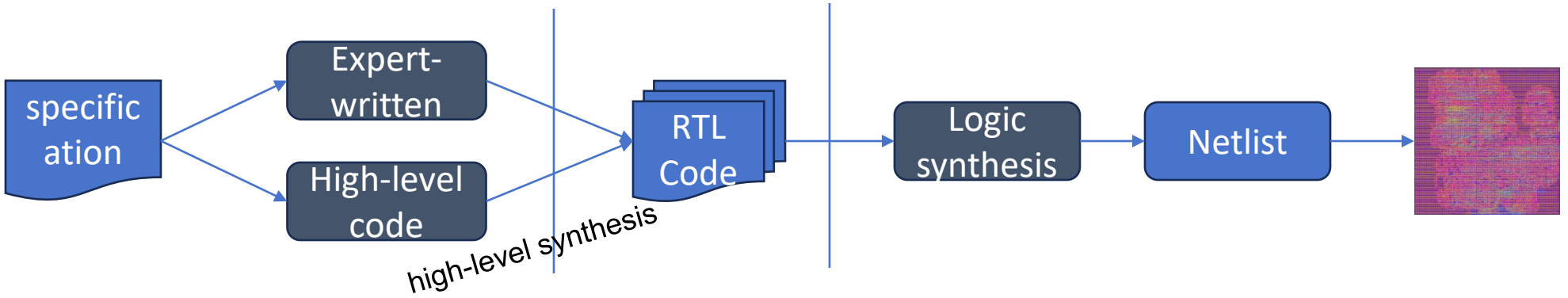


Reasoning

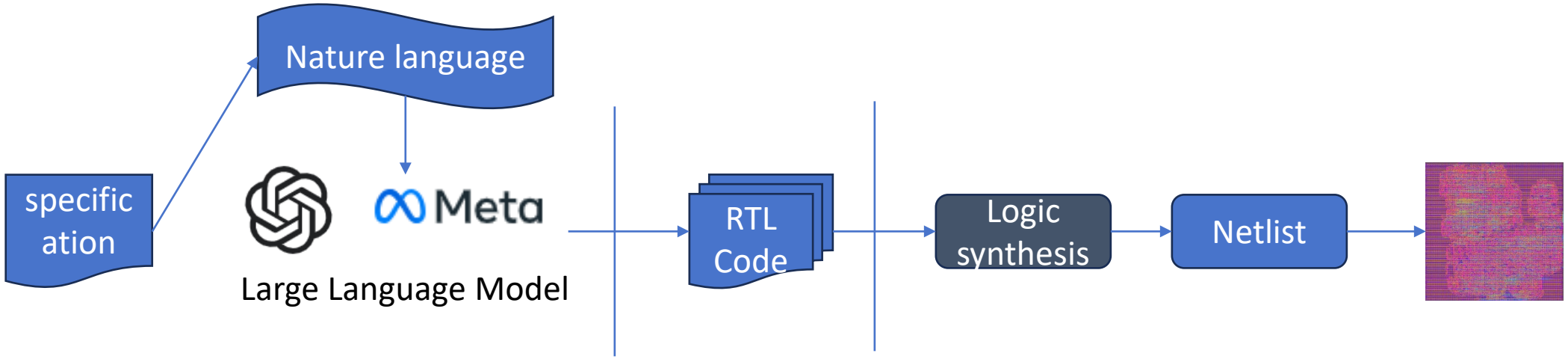
Large Language Model

University of Chinese Academy of Sciences

- Electronic Design Automation(EDA) flow



- LLM for hardware design



Risks of LLMs

- Fake news
- Malicious/Vulnerable code
- Sensitive content
- Private data leaks
- Fraud
- …

We need to embed watermarks to RTL code generated by LLM!

Prompt: Write a secure model that compares byte-based 64-bit user-given password to a stored 64-bit golden value.

```
module KeyComparator (
    input wire [7:0] key1,  // Input for the first secret key (8 bits)
    input wire [7:0] key2,  // Input for the second secret key (8 bits)
    output reg match         // Output indicating whether the keys match or not
);
    always @* begin
        // Initialize match signal to 1 (assume keys match)
        match = 1;
        // Compare each bit of the keys
        for (int i = 0; i < 8; i = i + 1) begin
            if (key1[i] != key2[i]) begin
                match = 0;
                break;
            end
        end
    end
Endmodule
```

Security vulnerability

Outline

中国科学院大学
University of Chinese Academy of Sciences

① Introduction

② Background

③ RTLMarker

④ Evaluation

⑤ Conclusion

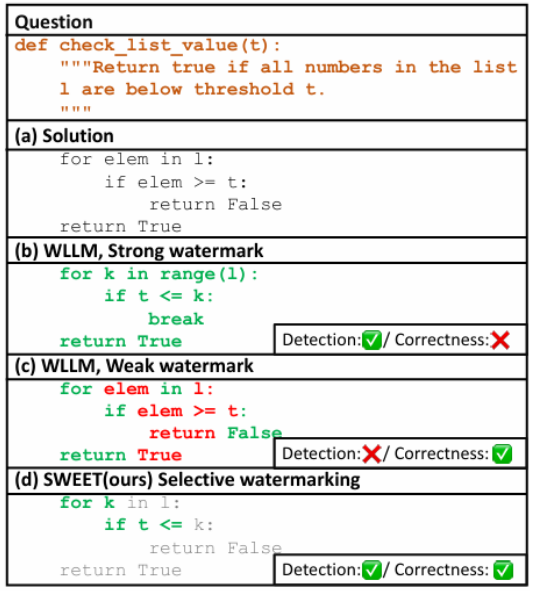
LLM Watermark

- Text Watermark: WLLM
- Code Watermark: SWEET

WLLM[1]

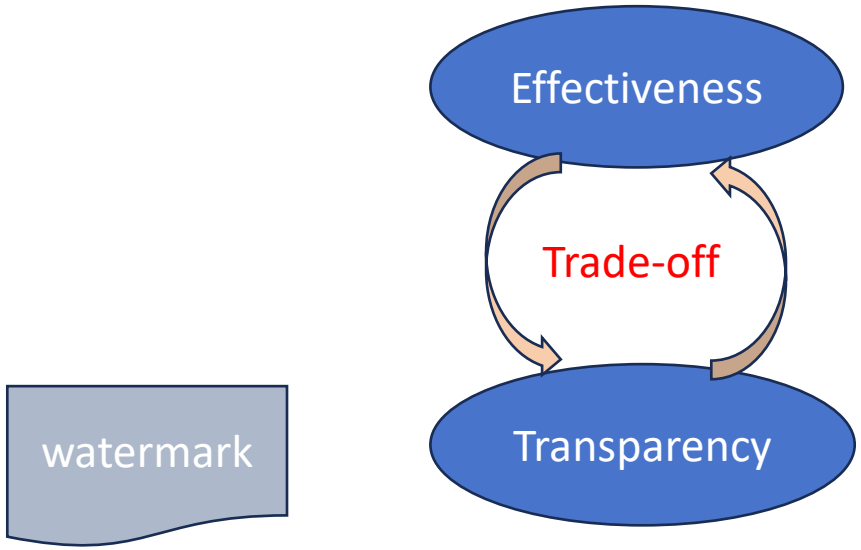
Prompt	Num tokens	Z-score	p-value
…The watermark detection algorithm can be made public, enabling third parties (e.g., social media platforms) to run it themselves, or it can be kept private and run behind an API. We seek a watermark with the following properties:			
No watermark Extremely efficient on average term lengths and word frequencies on synthetic, microamount text (as little as 25 words) Very small and low-resource key/hash (e.g., 140 bits per key is sufficient for 99.999999999% of the Synthetic Internet	56	.31	.38
With watermark - minimal marginal probability for a detection attempt. - Good speech frequency and energy rate reduction. - messages indiscernible to humans. - easy for humans to verify.	36	7.4	6e-14

SWEET[2]



[1] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, andTom Goldstein. 2023. A watermark for large language models. In InternationalConference on Machine Learning. PMLR, 17061–17084.
[2] Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun,Jamin Shin, and Gunhee Kim. 2023. Who wrote this code? watermarking for code generation. arXiv preprint arXiv:2305.15060 (2023).

LLM Watermark



The watermarks should be effectively embedded and detectable.

The watermark should preserve the code quality and remain inconspicuous.

The watermark should be resilient to common attack methods, such as string replacement attacks.

LLM Watermark

- Embedding watermarks to LLM-generated RTL code presents the following challenges:
  - Existing methods cannot guarantee the correctness of the watermarked RTL code.
  - There is a tradeoff between the transparency and effectiveness of watermarks.
  - Watermarks at the Register Transfer Level(RTL) are difficult to further embed into the synthesized netlist.

Outline

中国科学院大学
University of Chinese Academy of Sciences

① Introduction

② Background

③ RTLMarker

④ Evaluation

⑤ Conclusion

RTLMarker

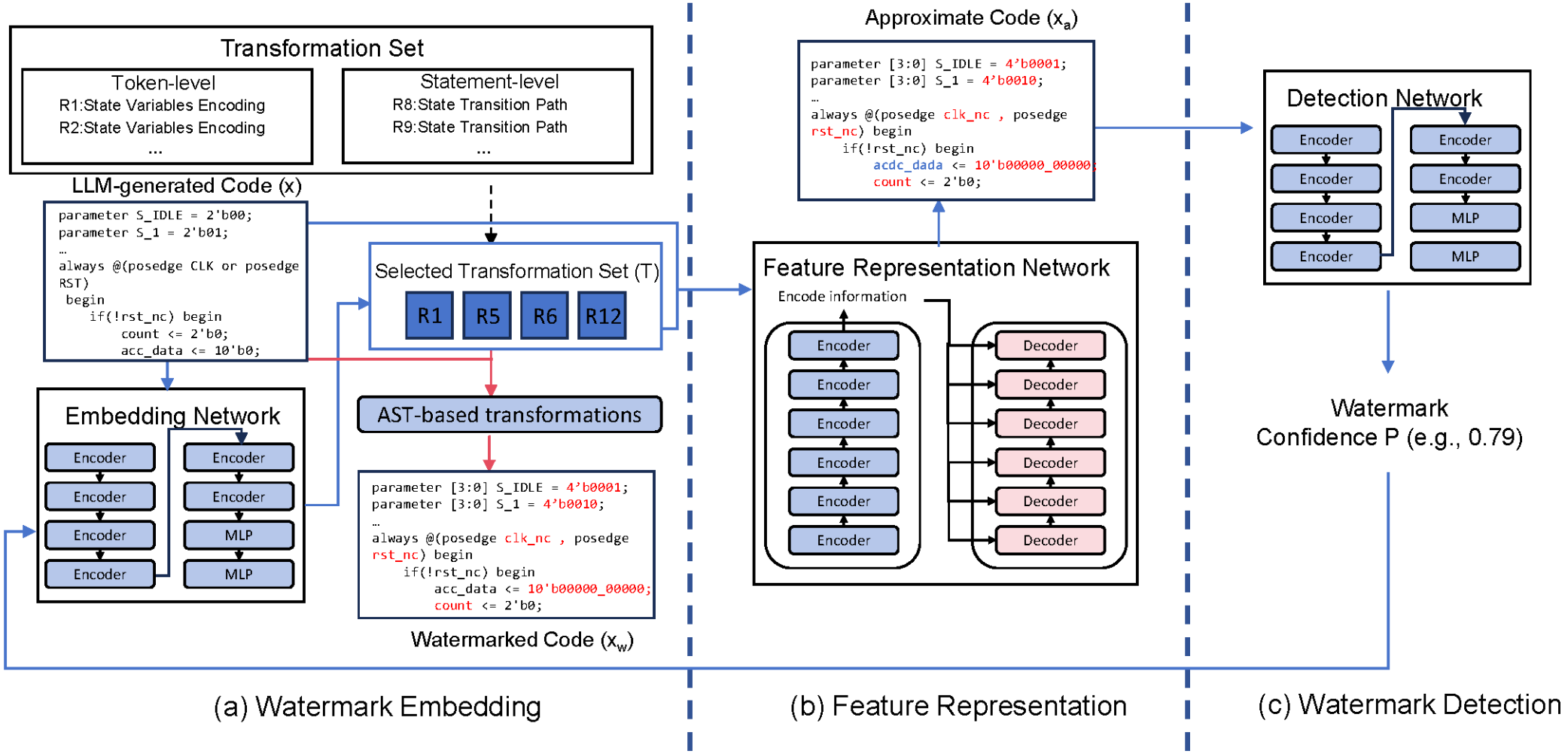
University of Chinese Academy of Sciences

- Rule-Based Verilog Code Transformations
  - Transformations can be split into Token level and statement level.
  - 15 code transformations are implemented by Pyverilog.

Granularity	Description	Transformation Rule
Token	State Variables Encoding	Using one-hot encoding instead of binary encoding, e.g., "RUN = 2'b10, STOP = 2'b11" to "RUN = 3'b010, STOP = 3'b100".
Token	Parameterized Module	Bit widths can be parameterized. e.g., `a[0:32]` and `b[0:32]` can be parameterized as `a[0:width]` and `b[0:width]`.
Token	Base Conversion	Binary, decimal, and hexadecimal can be converted to each other.
Token	Signal Sensitivity Formatting	Signal sensitivity lists are reformatted. e.g., "(posedge clk1 or negedge clk2)" to "(posedge clk1 , negedge clk2)".
Token	Bit Separation	Binary representations with many bits can include separators for better readability.
Token	Variable Name Replacement	Replace variable names, e.g., the signal name *rst* can be replaced with *rst_nc*.
Token	Bit Order	Bit order can be reversed, e.g., "watermark[0:N]" to "watermark[N:0]".
Statement	State Transition Path	Introduce specific transition sequences in the state transition path.
Statement	Combinational Logic Operation	AND gates, OR gates, and NOT gates can be converted into each other.
Statement	Combinational Logic Assignment	Add always@* during combinational logic assignment.
Statement	Ternary Operation	If-else statements can be expressed using ternary operations.
Statement	Signal Initialization Order	When initializing multiple signals, the order of initialization can be interchanged.
Statement	Add Comments	Add unique comments to some simple signals. e.g., "//Input signal clk_nc".
Statement	Conditional Statement Order	In the conditional statement "if(a & b)", the order of a and b can be interchanged.
Statement	Add Redundant Logic	Add redundant logic that does not impact the normal execution of the code.

RTLMarker Framework

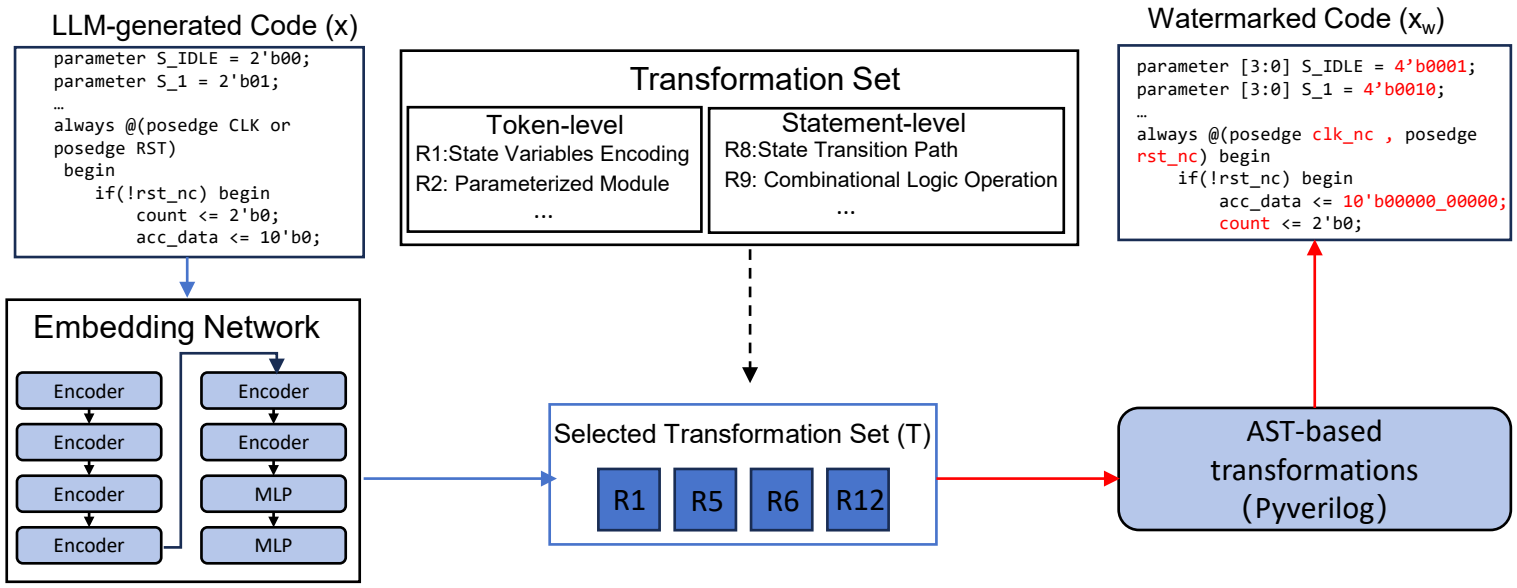
University of Chinese Academy of Sciences



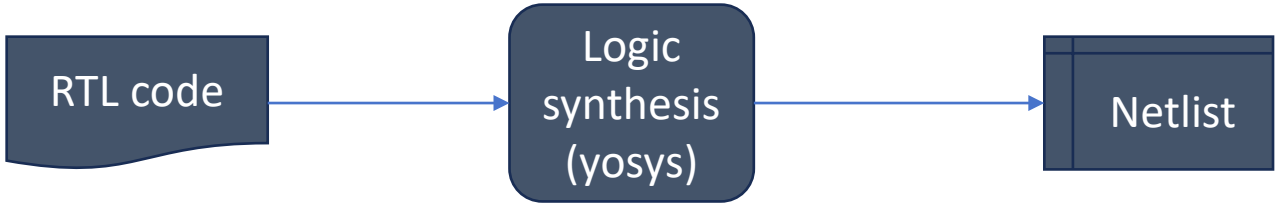
Watermarking Embedding

University of Chinese Academy of Sciences

- Learning-Based Watermark Embedding
  - The Embedding network outputs the selected Transformation Set based on the LLM-generated code.
  - An AST-based approach is used to apply the corresponding transformations, generating the watermarked code.

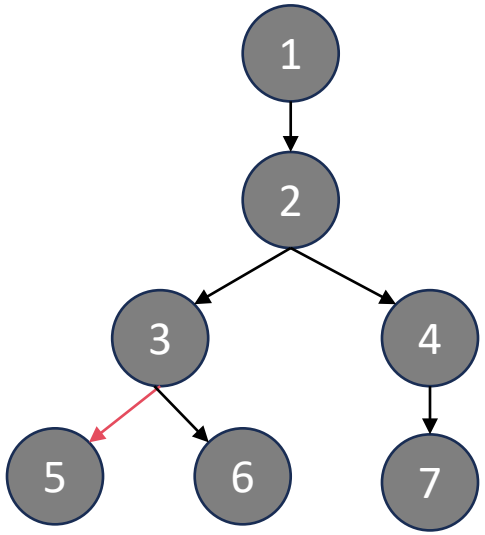


Watermarking Embedding



High-level semantic information will be lost

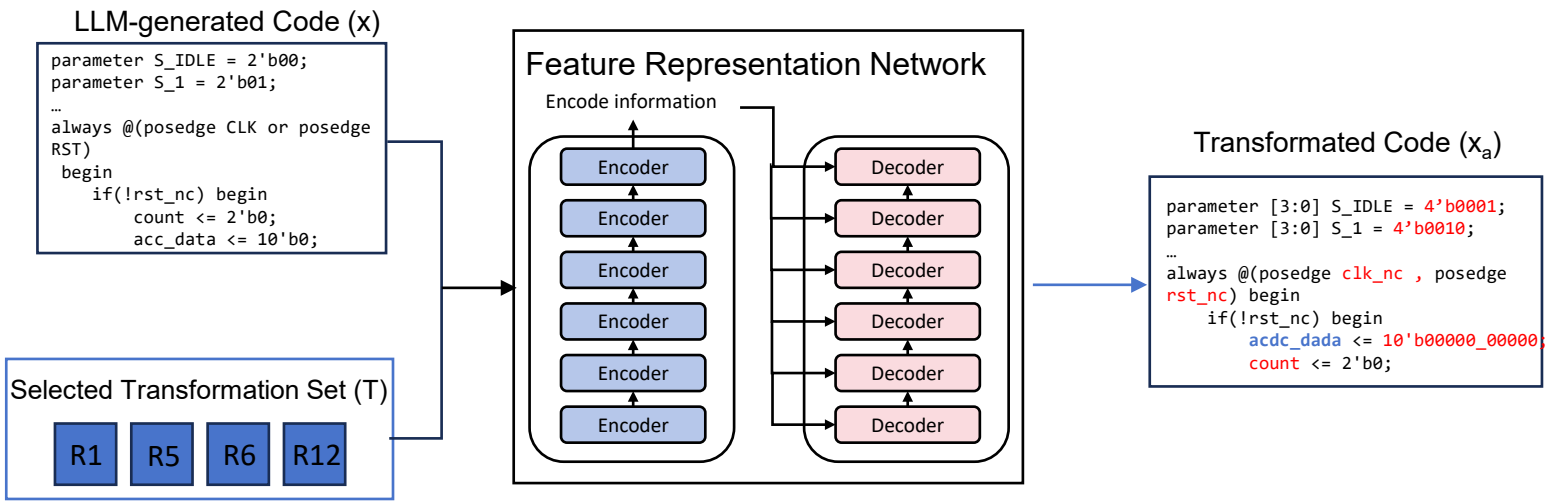
- Embedding watermark into netlist




```
module complex_data_processor (
    input wire clk,
    input wire reset,
    input wire watermark_trigger,
    output reg [7:0] processed_data
);
    reg [7:0] intermediate_data;
    always @(posedge clk) begin
        if (reset) begin
            processed_data <= 0;
            intermediate_data <= 0;
        end
        if (watermark_trigger) begin
            processed_data <= intermediate_data ^ 8'hA5;
        end else begin
            processed_data <= intermediate_data;
        end
    end
endmodule
```

Feature Representation

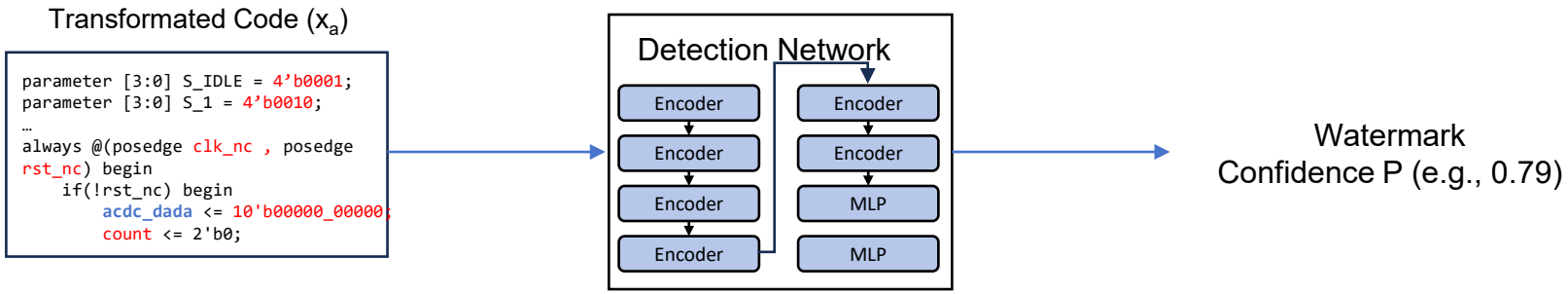
- Input:
  - LLM-generated Code(x) && Selected transformation set(T)
- Output
  - Transformed code(x_a)



Watermark Detection

University of Chinese Academy of Sciences

- Watermark Detection at the Register Transfer Level (RTL)



- Watermark Detection at Netlist Level
  - Employ synthesis tools yosys to synthesize code into netlist
  - Parse out the embedded watermark from netlist

Outline

中国科学院大学
University of Chinese Academy of Sciences

① Introduction

② Background

③ RTLMarker

④ Evaluation

⑤ Conlusion

Experiment Setup

中国科学院大学
University of Chinese Academy of Sciences

- Benchmark
  - RTLLM: 30 Verilog problems with varying levels of complexity.
  - VerilogEval: 156 verilog problems sourced from the Hdlbits website.
- Target Model
  - RTLCoder、GPT4、ChipGPT-FT
- Baseline
  - WLLM && SWEET
- Metrics
  - ACC: (TP + TN)/(TP + TN + FP +FN)
  - TPR: TP/(TP+FN)
  - FPR: FP/(FP+TN)

TP: True Positives
TN: True Negatives
FP: False Positives
FN: False Negatives

Evaluation

- Effectiveness
  - RTLMarker achieves an accuracy of over 95% on the RTLLM benchmark, while SWEET and WLLM only achieve 71.67% and 83.33%, respectively.
  - RTLMarker achieves an accuracy of over 92% on VerilogEval benchmark, while SWEET and WLLM only achieve 58.33% and 63.14%, respectively.
  - Compared to the VerilogEval benchmark, the accuracy of watermark embedding and detection is higher on the RTLLM benchmark.

Method	Benchmark	Model	ACC(%)	TPR(%)	FPR(%)
Ours(rtl)	RTLLM	GPT-4	96.67	93.33	0
		RTLCoder	95.00	90.00	0
		ChipGPT-FT	95.00	93.33	3.33
	VerilogEval	GPT-4	94.87	91.03	1.28
		RTLCoder	92.62	88.46	3.2
		ChipGPT-FT	92.95	89.74	3.85
SWEET	RTLLM	GPT-4	–	–	–
		RTLCoder	73.33	50	3.33
		ChipGPT-FT	71.67	46.67	3.33
	VerilogEval	GPT-4	–	–	–
		RTLCoder	58.65	20.51	3.21
		ChipGPT-FT	58.33	18.59	1.92
WLLM	RTLLM	GPT-4	–	–	–
		RTLCoder	83.33	70	3.33
		ChipGPT-FT	86.67	76.67	3.33
	VerilogEval	GPT-4	–	–	–
		RTLCoder	63.14	29.49	3.21
		ChipGPT-FT	64.74	32.05	2.56
Ours(netlist)	RTLLM	GPT-4	78.33	56.67	0
		RTLCoder	76.67	53.33	0
		ChipGPT-FT	76.67	53.33	0
	VerilogEval	GPT-4	62.18	24.36	0
		RTLCoder	62.82	25.64	0
		ChipGPT-FT	59.62	19.23	0

Evaluation

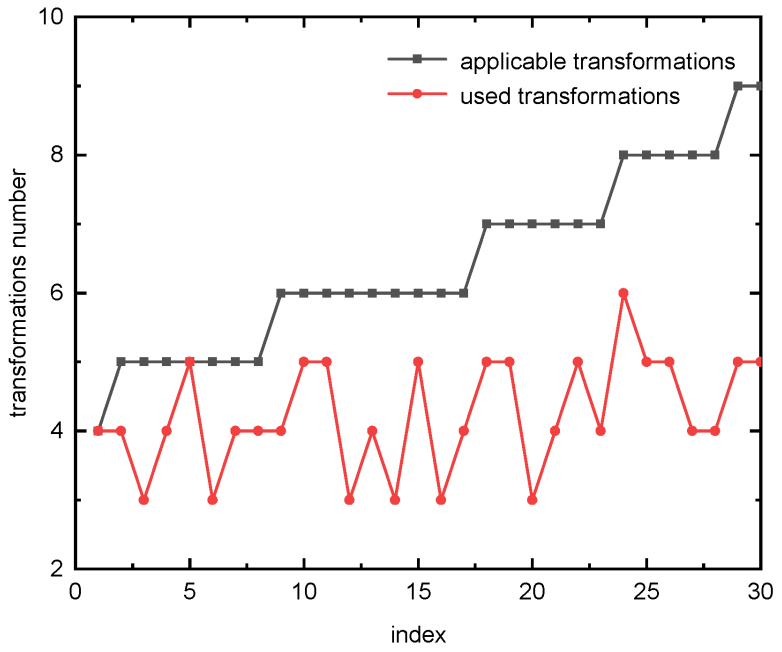
中国科学院大学
University of Chinese Academy of Sciences

- Robustness
  - Variable name replacement attack. We considered renaming 25%, 50%, 75% and 100% of the variables in the watermarked code.
  - RTLMarker is only slightly affected by variable name replacement attacks.

Attack	RTLLM			VerilogEval		
	ACC(%)	TPR(%)	FPR(%)	ACC(%)	TPR(%)	FPR(%)
Attack@25%	96.67	93.33	0	92.62	88.46	3.21
Attack@50%	96.67	93.33	0	90.38	86.53	5.76
Attack@75%	95.00	90.00	0	87.50	80.76	5.76
Attack@100%	91.67	83.33	0	80.45	70.51	9.62

Evaluation

- Transparency
  - We use the number of code transformations to measure the transparency of the watermark.



The average number of applicable code transformations in the RTLLM benchmark is 6.42, while the number of code transformations that RTLMarker utilizes is 4.25, effectively enhancing the transparency of the watermark

Outline

中国科学院大学
University of Chinese Academy of Sciences

① Introduction

② Background

③ RTLMarker

④ Evaluation

⑤ Conclusion

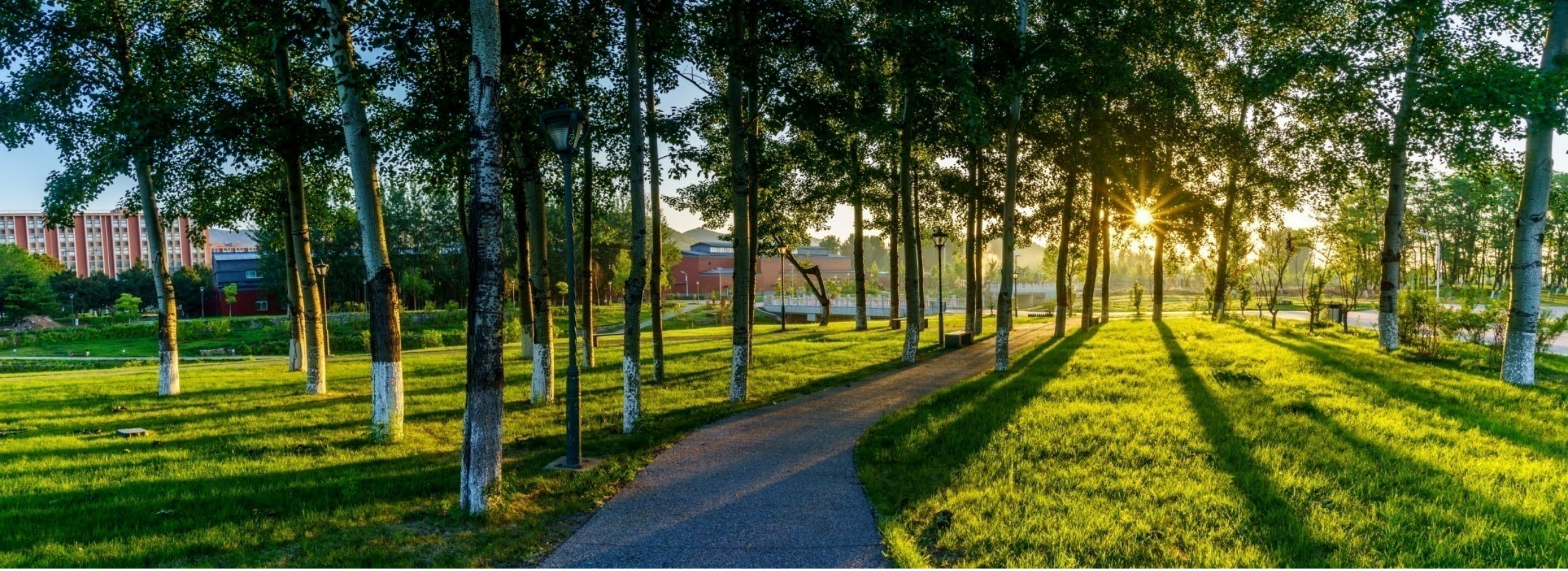
Conclusion

中国科学院大学
University of Chinese Academy of Sciences

- To our knowledge, this research is the pioneering effort to introduce a practical and efficient watermarking framework designed to safeguard the copyright of RTL generated by large language models.

- We propose a comprehensive suite of Verilog-centric code transformations and concurrently create a state-of-the-art tool powered by Pyverilog to facilitate these transformations.

- Our study introduces an advanced framework for embedding and identifying hardware watermarks, functional at both the Register Transfer Level (RTL) and the logic netlist level.



THANKS

中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

中国科学院大学
University of Chinese Academy of Sciences

智能计算机研究中心
Research Center for Intelligent Computing Systems