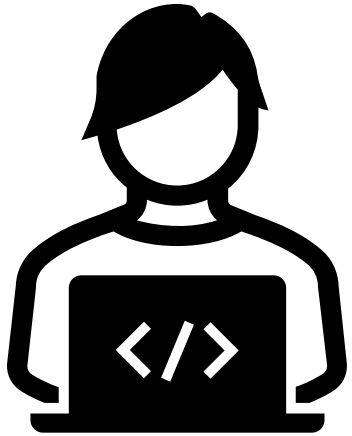
基於 LLM 的 RTL 程式碼修正與驗證輔助系統

AI-Assisted RTL Code Repair and Verification System

組別：E-13　　指導教授：許舒涵　　專題成員：楊柏恩

專題背景與動機

- RTL 開發常見問題：
  - 語法錯誤
  - 介面不一致
  - 程式結構不清楚
  - 修改後功能行為改變



- 動機：

  -人工檢查耗時，且容易遺漏細節。

  -本專題希望建立一套 AI 輔助 RTL 修正與驗證流程，使模型能根據工具回饋反覆修正，自動化Verilog的開發與設計過程。

專題內容與技術特色

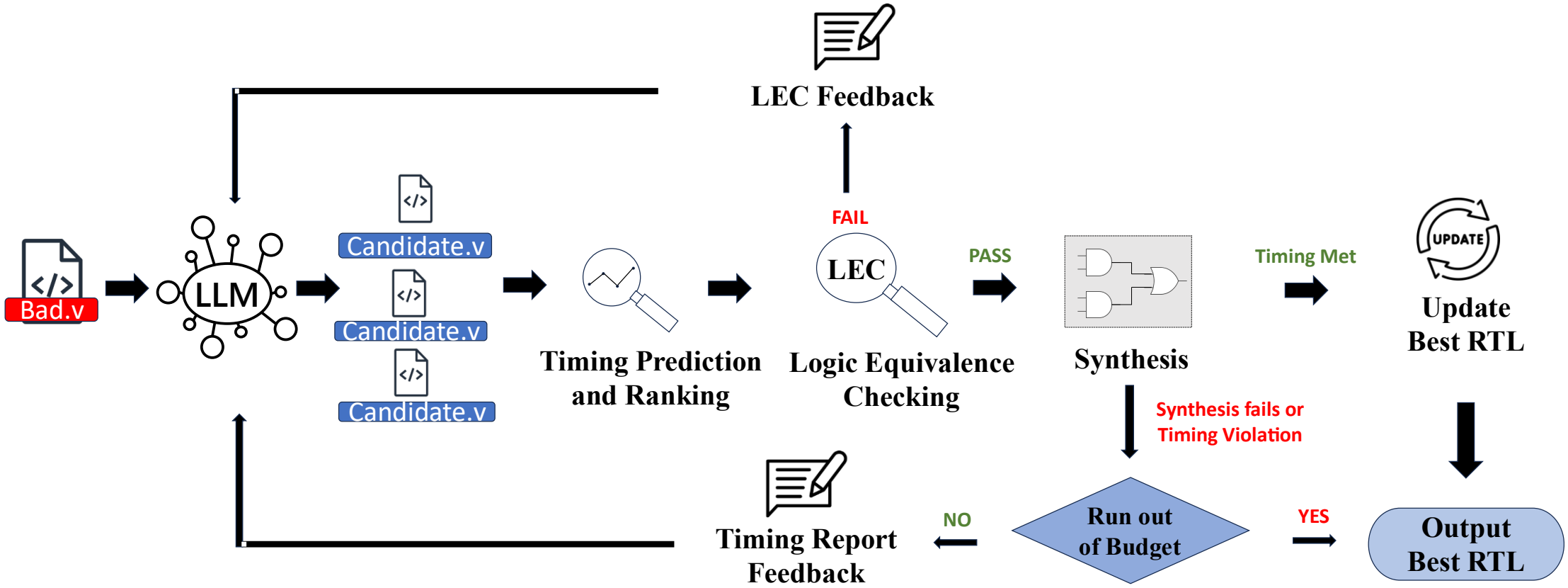


圖1. 系統架構圖

專題內容與技術特色（Cont.）

- **Fine-tuned Model：**
  - 開源難以找到現成 Bad Code Training Dataset。
  - 人工+LLM輔助改寫並標註Source Code中功能壞掉或難以maintain的部分，以及如何修正。
- **LEC邏輯等價檢查：**
  - 利用yosys將原始RTL與改寫後RTL轉成內部邏輯網路，再透過邏輯化簡與等價檢查流程，比較兩個設計是否具有相同的輸入輸出行為。
- **Feedback擷取和標準化：**
  - 將合成工具（如：Design Compiler）的 Report 透過 Python 程式碼擷取出利於 LLM 吸收的形式，過濾不必要雜訊。
- **完整Closed-Loop系統架構：**
  - 可利用socket將模型產出與工具回饋在不同通訊設備上交換，合成使用之pdk檔案和LLM token只在local上被載入，不會跟著交換。
  - 生成->合成->反饋->優化迭代。透過 Python 程式碼控制，可自由調整迭代次數、生成候選數量、合成Constraint等。

成果展示

```
------------------------------------------------
🚀 [Baseline] case=flipflop | top=flipflop
------------------------------------------------
🏭 先對初始 bad RTL 做 baseline synthesis...
📊 baseline WNS = 2.03

>>> [Iteration 1] 產生 3 個候選 RTL（無 feedback）
    [GPT-5.4] approx prompt tokens = 302, rtl lines used = 1
    🔍 開始對 cand_1 跑 LEC...
    🔍 cand_1 的 LEC 已結束: LEC Passed
    ✅ cand_1 通過 LEC
    🔍 開始對 cand_2 跑 LEC...
    🔍 cand_2 的 LEC 已結束: LEC Passed
    ✅ cand_2 通過 LEC
    🔍 開始對 cand_3 跑 LEC...
    🔍 cand_3 的 LEC 已結束: LEC Passed
    ✅ cand_3 通過 LEC
    ℹ️ 本輪最多送 2 個 LEC-passing candidates 去 synthesis。
    🏭 對 cand_1 執行真實 synthesis... (used 0/6)
    📊 cand_1 真實 WNS = 3.69
    🏭 對 cand_2 執行真實 synthesis... (used 1/6)
    📊 cand_2 真實 WNS = 3.69
✅ 更新目前最佳 RTL，來源 cand_1，best WNS = 3.69
🎉 找到 WNS >= 0 的候選，終止此 case。
```

圖2. 系統運作圖(成功案例)

```
------------------------------------------------
🚀 [Baseline] case=reg_32 | top=reg_32
------------------------------------------------
🏭 先對初始 bad RTL 做 baseline synthesis...
📊 baseline WNS = 3.67

>>> [Iteration 1] 產生 3 個候選 RTL（無 feedback）
    [GPT-5.4] approx prompt tokens = 242, rtl lines used = 1
    🔍 開始對 cand_1 跑 LEC...
    🔍 cand_1 的 LEC 已結束: ERROR: syntax error, unexpected
    ❌ cand_1 LEC 失敗
    🔍 開始對 cand_2 跑 LEC...
    🔍 cand_2 的 LEC 已結束: ERROR: syntax error, unexpected
    ❌ cand_2 LEC 失敗
    🔍 開始對 cand_3 跑 LEC...
    🔍 cand_3 的 LEC 已結束: ERROR: syntax error, unexpected
    ❌ cand_3 LEC 失敗
⚠️ 本輪沒有任何候選通過 LEC，直接進入下一輪。
    LEC failures:
      [cand_1] ERROR: syntax error, unexpected TOK_REG, expec
      [cand_2] ERROR: syntax error, unexpected TOK_REG, expec
      [cand_3] ERROR: syntax error, unexpected TOK_REG, expec
```

圖3. 系統運作圖(失敗案例)

成果展示

表一：在100筆測試資料集中，模型成果比較。

模型	Rewrite 通過 LEC	無法合成 -> 可合成	整體說明
GPT5.4	49	13	先進商用大型模型
llama3.1-8B-Instruct	13	4	大型開源模型
llama3.1-70B-Instruct	15	5	大型開源模型
CodeGemma	4	0	大型開源模型
BaseModel	25	1	RTL-Coder(Mistral)
MyModel	46	12	本專題 fine-tuned 模型

與Fine-Tune前相比，效果顯著提升，並且使用7B小模型達到接近大型商用模型的成果