National Tsing Hua University
Department of Electrical Engineering
EE4292 IC Design Laboratory (積體電路設計實驗)
Fall 2021

# CVAE-seq2seq Robotic Arm Control AI accelerator

Team Member: 鍾宇騫(107062205)、黃韋珀(107070032)

## 1. Functionality

我們的 CVAE seq2seq decoder 為課程前就已訓練好的模型，但僅有 Pytorch model 和 weight 的 ckpt 檔，project 內的 C++軟體模型、RISCV 的驗證 C 程式、RTL code 都是為了 project 寫的。

該模型可以根據外界輸入的手臂狀態和目標點位置，迭代運算出每一步的移動軌跡，使機械手臂的手掌移動到目標點[1]。如 Figure.1 所示，CVAE decoder 主要由三個 Fully Connection 和一個 Gated Recurrent Unit (GRU)的運算所組成。一開始的 *rand* 值會先經過一層 Fully Connection (*FC_l2h*)計算出 GRU 的初始 *hidden state*，之後 GRU 便會不停迭代運算出下一層的 *hidden state*，供後半部的 Fully Connection 計算出手臂移動軌跡 *state* 和判斷是否完成工作的 label *config*。

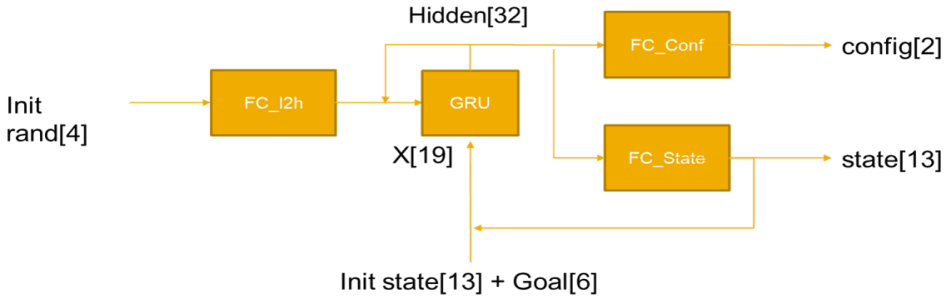


Figure.1 CVAE seq2seq decoder

由 Equation.1 的 GRU 計算公式可知其中的矩陣相乘計算原理與 Figure.1 中的 Fully Connection 是一樣的，並且因為 SRAM 無法合成進 Design，Weights 輸入的頻寬有所限制，因此在電路架構的設計我們只放置了一個 Fully Connection 的模組來重複使用，以降低 I/O pin 的數量。

$$R_t = sigmoid(W_{ir}X_t + b_{ir} + W_{hr}h_{t-1} + b_{hr})$$
$$Z_t = sigmoid(W_{iz}X_t + b_{iz} + W_{hz}h_{t-1} + b_{hz})$$
$$N_t = tanh(W_{in}X_t + b_{in} + R_t(W_{hn}h_{t-1} + b_{hn}))$$
$$h_t = (1 - Z_t)N_t + Z_t h_{t-1}$$

Equation.1 GRU 計算公式

而在 Sigmoid 與 Hyper Tangent 的實作上，我們嘗試在 C++上以查表(LUT)來模擬，但 LUT 的誤差於計算過程中不斷累加，使得最終結果與 Pytorch 的運算結果相差過大，並且這部份我們在 Webots 上有實際去模擬算出來的軌跡是無法移動到目標點的。因而我們採用 CORDIC 演算法[2]來實作這兩種自然對數運算。如 Figure.2 & 3 所示，該演算法可以透過對 Input 和 Output 做加法和邏輯運算的處理，並使用相同的 data path 來實現 Sigmoid 和 Hyper Tangent。且因為 CORDIC 是使用逼近的方式來迭代計算結果，我們參考論文[2]調整出適合 CVAE decoder 運算資料範圍的參數，以 latency 41 cycles,

throughput 1 result/cycle 的架構實作於 CVAE decoder 內。

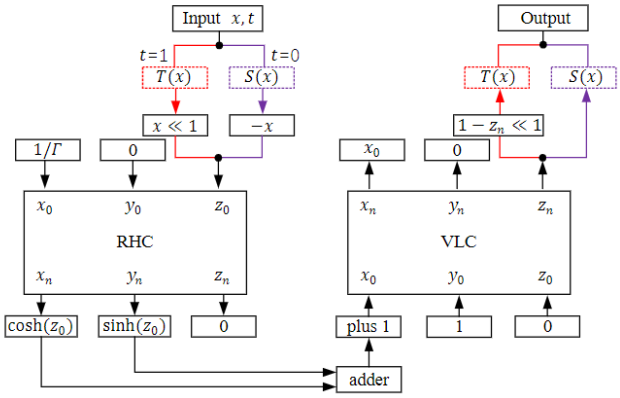


Figure.2 CORDIC based sigmoid/tanh algorithm [2]

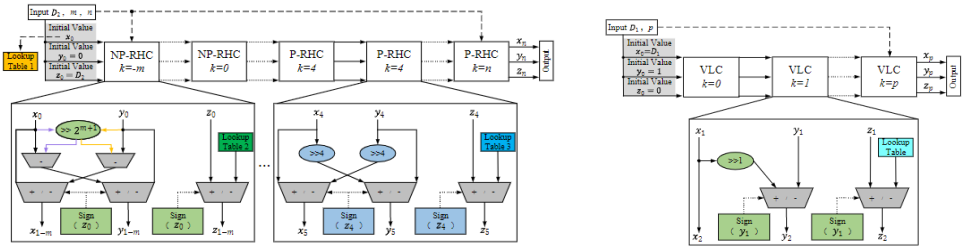


Figure.3 Pipelined RHC (left) and VLC (right) module [2]

如 Figure.4，電路架構的設計我們分為三部份: Control、Storage、Data path。整個系統由 Control 電路的 FSM 來控制，所有 Input 控制訊號都會送到 FSM，再由 FSM 來控制其他電路的行為。

Storage 的部分因為無法合成 SRAM 以及為了降低 I/O pin 的數量，我們開了一組 Register files 供內部運算的資料暫存，總共有 7,840 bits。Register files 會負責接收內部運算好的資料或是外部進來的資料，再透過 FSM 將資料送至其他的 data path。

Data path 的部分有兩個: 一為 Fully Connection，由於上述提及 **SRAM 無法合成**，所以我們只配置了一個 Fully Connection 電路，並且受限於 Weight 頻寬，運算速度為 1 MAC/cycle; 二為 GRU 電路。因為 **Register Files 沒有如外部 SRAM 的頻寬限制**，GRU 的設計為完全的 Pipeline，Latency 為 89 cycles，Throughput 為 1 result/cycle。GRU 其中的 Hyper Tangent 和 Sigmoid 我們是根據 Figure.2 來實作，並且用到的參數是以 LUT 的方式刻在電路內。

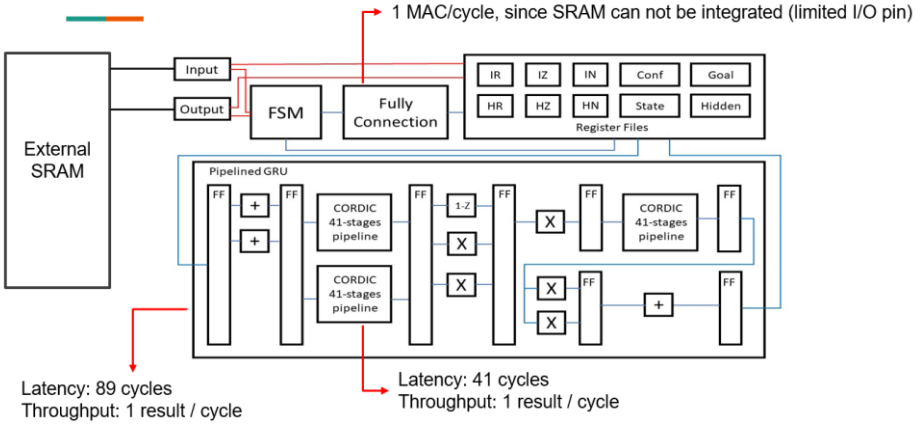


Figure.4 CVAE decoder top module

整體的 Data flow 為 Testbench 透過 control 訊號通知 CVAE decoder 資料已經準備好，CVAE decoder 會自行向外部 SRAM 讀寫並運算結果，讀進來的資料和運算結果都會暫存於 Register files，而在運算結果都寫回外部 SRAM 後，會送一個 finish 訊號給 Testbench 來開始驗證結果。

# 2. Specification

## (a) Overall PPA

| | Spec | Our result |
|---|---|---|
| Quantization RMSE | < 0.05 | 0.03 |
| Clock period | < 6.4 ns | 5.8 ns (post sim) |
| Iteration latency | < 10,000 cycles | 5,454 cycles |
| Power | < 100 uW | 29 uW |
| Area | < 500,000 $um^2$ | 1,001,104 $um^2$ |

## (b) Clock period

| Unit: ns | Synthesis | P&R | Post sim |
|---|---|---|---|
| Clock period | 5.12 | 5.6 | 5.8 |

## (c) Power profiling

### (1) Overall

| Unit: uW | DC | ICC | PT (pre-sim) | PT (post-sim) |
|---|---|---|---|---|
| Leakage power | 4810 | 5400 | 5600 | 5400 |
| Dynamic power | 22400 | 22900 | 5400 | 23500 |
| Total power | 27300 | 28300 | 11000 | 29000 |

### (2) Post simulation with Prime Time

由表可知，絕大部分的 Power 都是在 clock 上，data path 與 storage 的功耗其實占比不高。

| Module | Switching | | Internal | | Leakage | | Total | |
|---|---|---|---|---|---|---|---|---|
| Unit | uW | % | uW | % | uW | % | uW | % |
| Clock network | 17200 | 95.0 | 4620 | 85.0 | 265 | 4.9 | 22100 | 76.3 |
| Registers | 32 | 0.2 | 26 | 0.4 | 2650 | 49.1 | 2710 | 9.4 |
| Combinational | 874 | 4.8 | 791 | 14.5 | 2480 | 46.0 | 4150 | 14.3 |

### (3) Gate level simulation with Prime Time

從 Gate level sim 看各 module 功耗可以發現耗能與面積成正比(參考面的 Area profiling)。

| Module | Switching | | Internal | | Leakage | | Total | |
|---|---|---|---|---|---|---|---|---|
| Unit | uW | % | uW | % | uW | % | uW | % |
| FC | 85.8 | 61.7 | 416 | 1.9 | 161 | 3.3 | 664 | 2.4 |
| GRU-sigmoid 0 | 9.15 | 6.9 | 4060 | 18.2 | 917 | 19.1 | 4990 | 18.3 |
| GRU-sigmoid 1 | 11.2 | 8.0 | 4070 | 18.2 | 918 | 19.1 | 5000 | 18.3 |
| GRU-tanh | 15.8 | 11.3 | 4080 | 18.2 | 904 | 18.7 | 5000 | 18.3 |
| GRU-others | 4.35 | 3.0 | 290 | 1.3 | 241 | 5.0 | 510 | 2.0 |
| FSM & Register | 13.0 | 9.2 | 9480 | 42.3 | 1670 | 34.7 | 11100 | 40.7 |
| Total | 140 | 100 | 22400 | 100 | 4810 | 100 | 27300 | 100 |

(d) Area profiling

面積主要由 GRU 的三顆 CORDIC 電路和 Register files 所組成。Total cell 的面積是有符合 spec 的，但因為 APR 時把 core utilization 設 0.5 或更高會繞不出來，我們推測問題是在於 Register files 的 bit 數過多(7840 bits)

| Module | Absolute area | Percentage |
|---|---|---|
| Fully Connection | 11,615 $um^2$ | 3.9 % |
| GRU-sigmoid 0 | 55,479 $um^2$ | 18.9 % |
| GRU-sigmoid 0 | 55,586 $um^2$ | 18.9 % |
| GRU-tanh | 54,362 $um^2$ | 18.5 % |
| GRU-others | 18,009 $um^2$ | 6.1 % |
| FSM & Register files | 99,223 $um^2$ | 41.5 % |
| Total cell area | 294,277 $um^2$ | 100.0 % |
| APR area | 1,001,104 $um^2$ | Core utilization: 0.3 |

# 3. Implementation

(a) 演算法模型

已根據課程前已有的 Pytorch 的模型演算法實作出 C++版本的軟體程式，該軟體可以基於 Pytroch 的浮點數資料產生出 Quantization 後的 Weight 和 Pattern。並且能透過快速的更改單一變數來調整 CORDIC 演算法和 Quantization 的各項參數來達到電路的運算精度需求。

(b) 系統架構設計

如 Figure.5 以及上述的架構解釋，設計出整個電路的架構，並將 Module 的 I/O 與功能開好規格分配給組員。

(c) 浮點數 Quantization

採用 32-bits fixed point 來實作，調完參數後的 RMSE 為 0.03，滿足規格

(d) 電路設計

根據(b)的系統架構，組員各自設計分配的部分，並在設計完成後組裝起來與 TB 測試。

(e) 電路合成& APR

參考 Lab10 的 Script，沒有設定 clock gating，並修改部分 APR 參數，如 core utilization。

# 4. Verification

(1) Performance

為了驗證加速器的加速效果，我們使用 Xilinx vcu118 FPGA 合成包含 RISCV Araine 64-bits CPU 的完整系統，在其運行 CVAE Decoder 的軟體運算，並在軟體內使用 Assembly code 量測計算所需的 cycle count 來與我們加速器的運算效果比較。這個 FPGA 的驗證環境是課程前就有的，但在上面跑的 C 程式是為了這份 Project 寫的。

Table.1 Accelerator and software performance

| | Iteration cycle count | Clock period |
|---|---|---|
| Accelerator | 5,454 | 5.8 ns |
| SW in RISCV | 519,611 | 6.4 ns |

根據 Table.1 結果，可得到相較軟體的加速效果達 **95 倍**。

$$\text{Speed up} = 519{,}611 \div 5{,}454 = 95.27$$

(2) Functionality

我們設計的 CVAE decoder 是根據真實手臂的參數所訓練的，因此我們使用機械元件 3D 模擬軟體 Webots 內的七軸機械手臂模型來驗證我們電路所產出的 Output 是否能真的達到目標，也就是給定機械手臂初始狀態、目標三軸座標和三軸方向，電路是否能計算出每個 iteration 的軌跡，使得 Webots 內的機械手臂可以將手掌中心移動到影片中黃色的目標點。驗證方式是先產生好 input pattern 存成輸入檔案，給 Ncverilog 模擬後寫成 output 檔案，再把這些 output 檔案放到 Webots 上執行。並且 Webots 的模擬環境是課程前就有的。

Demo Video 1: https://www.youtube.com/watch?v=3Of2RPVDa7U

多筆 Testcases，每筆 Testcase 中，手臂會根據電路已運算好的輸出移動到目標點。

Demo Video 2: https://www.youtube.com/watch?v=gNhx5yFs1H0

單筆 Testcase，手臂會根據每個 iteration 的軌跡移動，每個 iteration 中加了 1 second sleep time 來方便觀察。

# 5. Contribution

(1) 鍾宇騫

- 演算法研究& 軟體模型實作
- Testbench 驗證環境撰寫
- 電路架構設計
- GRU 電路設計& Modules 整合
- 電路合成 & APR
- RISCV 驗證環境架設 & 效能分析

(2) 黃韋珀

- 演算法研究&軟體模型實作
- Testbench 驗證環境撰寫
- Fully Connection 電路設計& Modules 整合
- 電路合成 & APR

# 6. Reference

[1] Long-horizon Trace Planning with CVAE-Seq2Seq Variational Inference for Robot Manipulation, master's thesis of department of Electrical Engineering, NTHU

[2] A CORDIC-Based Architecture with Adjustable Precision and Flexible Scalability to Implement Sigmoid and Tanh Functions, IEEE International Symposium on Circuits and Systems (ISCAS 2021)