國立清華大學 電機工程學系

實作專題研究成果報告

Implementation of Application Specific Instruction set Processor Design

專用指令集處理器設計實作

專題領域：系統領域

組　　別：B493

指導教授：劉靖家 教授

組員姓名：鄭昇明、羅谷安、林汶昇

研究期間：113 年 9 月 27 日 至 113 年 11 月 3 日止，共一個月

摘要

本研究旨在探討專用指令集處理器(Application Specific Instruction set Processor, ASIP)的設計與實作。我們利用 Synopsys 提供的處理器設計工具與設計流程，針對 MLperf Tiny 這個專為嵌入式設備對於 AI 運算所設立的基準(Benchmark)，進行 RISC-V 處理器設計與優化。我們通過 SIMD(Single Instruction Multiple Data)、特殊指令(Special Instruction)如 MAC(Multiply-Accumulate)等觀念進行擴展指令集(Extended Instructions)設計，並在硬體層面做出相應的調整，並使用工具進行指令集模擬(Instruction Set Simulation, ISS)。實驗結果顯示，修改過後的處理器在 AI 運算所需的週期數(Cycle count)、指令數(Instruction count)皆獲得了顯著的減少。

1. 前言

1-1. 研究動機與目的

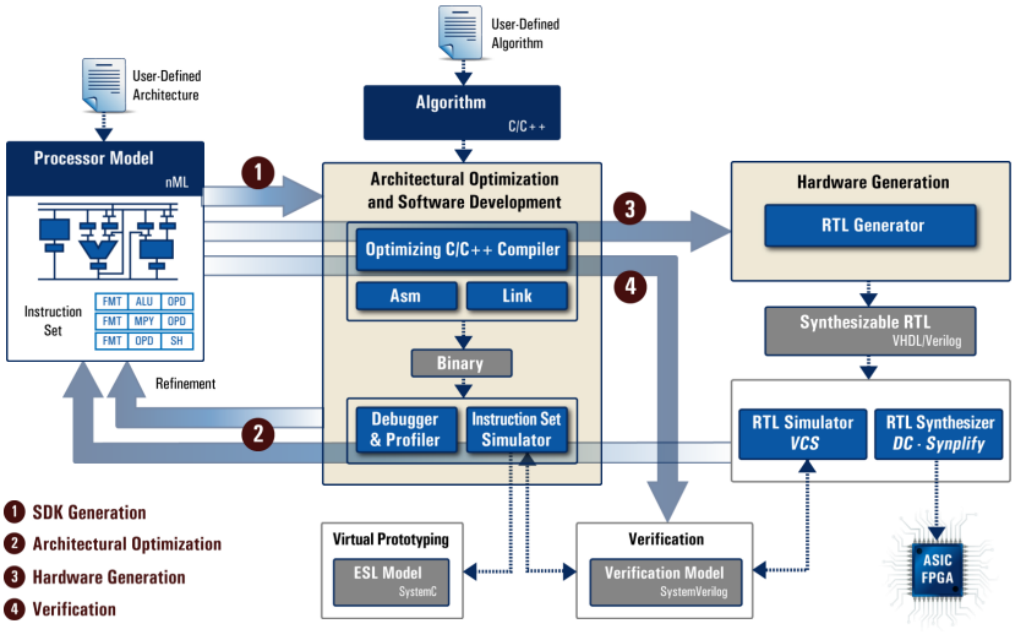
隨著近期各種應用領域使用機器學習的需求增加，機器學習的計算量與複雜度也隨之增加。使用專用指令集處理器(Application Specific Instruction-set Processors, ASIP)，為了特定應用提供專用的指令集架構，為嵌入式設備加速機器學習的解決方案之一。其他的解決方案有如：通用處理器(General Purpose Processor, GPP)，編程的靈活度較高且價格較便宜，但效能較差且較耗能；硬體加速器(Hardware Accelerator)，效能較好且較不耗能，但是缺乏編程的靈活度；以及圖形處理器(Graphics Processing Unit, GPU)，相對於 GPP 來說效能較好，但也更加耗能。而 ASIP 則介於效能與編程的靈活度兩者的極端之間。我們希望透過實作 ASIP 的設計流程，針對特定 AI 運算進行設計與優化，提升處理器的效能(Performance)。

1-2. 文獻探討

我們將介紹 Synopsys ASIP Designer 的設計流程和工具，並介紹其所提供的 RISC-V 範例模型，我們將在此模型之上進行修改與設計。以及介紹 MLperf Tiny 這個專們針對嵌入式系統 AI 運算的基準。

1-2-1. ASIP 設計流程

Synopsys ASIP Designer 為處理器設計提供了一套更高效的設計流程，有別於傳統的處理器設計方法，ASIP Designer 更方便的連結了軟硬體與設計驗證，加快了處理器設計的佚代速度。

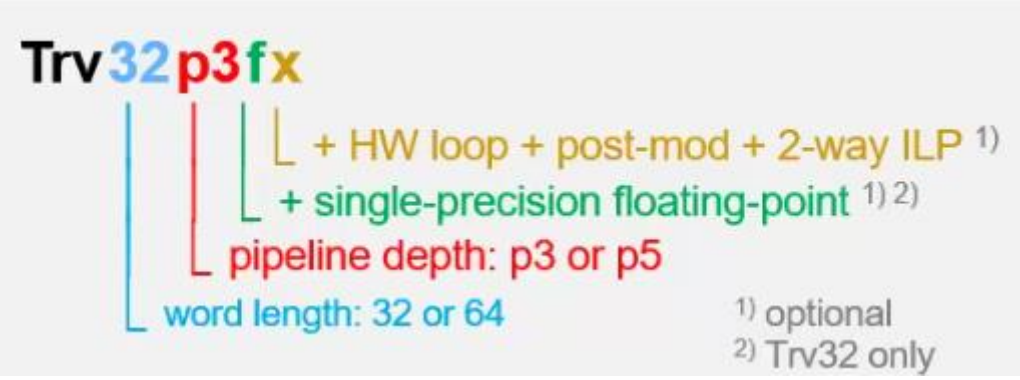


圖一、Synopsys ASIP Designer 設計流程

如圖所示，我們使用 nML 語言，一個描述處理器架構和指令集的高階硬體描述語言，來定義處理器的硬體架構和指令集行為。而後，設計工具會連結軟體(C/C++)，生成所需的軟體開發工具(Software Development Kit, SDK)，包含了編譯器(Compiler)、組譯器(Assembler)、連結器(Linker)等，並編譯出二進位碼，接下來就可由工具進行指令集層級的模擬，以及除錯和分析(Profiling)，再回歸設計階段，以此進行佚代。除此之外，還可以利用工具將 nML 轉換成 RTL(Register Transfer Level)語言，進行後續的 RTL 模擬、合成(Sythesis)等。最後，還可以利用工具去生成 System C 的 ESL(Electrical System Level)模型進行驗證等等。不過在本研究中，我們暫不討論 RTL 和 ESL，我們僅止步在利用 ASIP Designer 所提供的工具進行 nML 設計和指令集層級模擬並分析效能，包含週期數與指令數，面積、操作頻率、功耗等問題並不考慮在內。

1-2-2. Trv 系列處理器介紹

Trv 系列處理器是 ASIP Designer 所提供之範例，支援 RISC-V 指令集。本研究將會以此出發，進行擴展指令集處理器設計。



圖二、Trv 系列處理器命名方式與意義

Trv 系列處理器的命名規則，在 trv 之後的字串，依序代表處理器位元數、pipeline 級數、是否支援浮點數運算、是否支援通用加速功能。以我們所要修改的 trv32p3x 為例，它支援 RV32IM 指令集，有著三級 pipeline、不支援浮點數運算、支援加速功能如兩排指令層級並行(2-way Instruction-level Parallelism, ILP)、post-modify 讀寫、zero overhead hardware loop 等功能。

1-2-3. MLperf Tiny benchmark

MLperf 是一個專為硬體 AI 運算所建立的基準，這裡我們著重介紹針對嵌入式 MLperf Tiny。MLperf Tiny 以 Tensorflow Lite 為訓練框架(Training framework)，將資料集經過訓練，儲存為 graph format，共有四個不同種類的 AI 應用。而嵌入式設備將對這四個 AI 應用的運算進行效能評估。在本研究中，我們僅透過擴展指令集這個方式，從硬體方面提升效能，並保證運算結果的一致性。

表一、MLperf Tiny 介紹

應用	功能簡介	資料集	Graph format	模型
Keyword Spotting	小單字庫關鍵詞檢測	Speech Commands	*kws_ref_model_int8.tflite*	DS-CNN
Visual Wake Words	二元圖像分類	Visual Wake Words Dataset	*vww_96_int8.tflite*	MobileNet
Image Classification	小型圖像分類	Cifar10	*pretrainedResnet_int8.tflite*	ResNet
Anomaly Detection	偵測機械運轉聲之異常	ToyADMOS	*ad01_int8.tflite*	Deep AutoEncoder

2. 研究方法

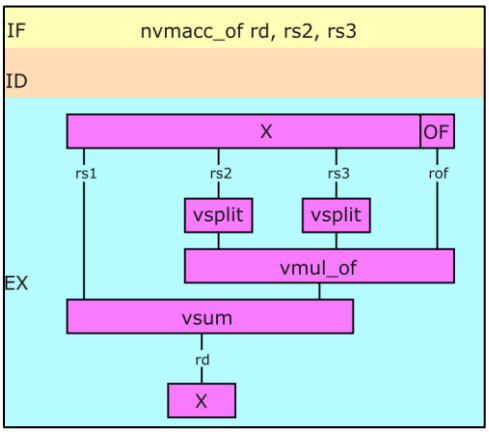
2.1 擴展指令集設計

我們觀察到 ConvPerChannel 的程式碼，MAC 運算是使用 32-bit 進行運算，而 MAC 所使用的變數最初是以 8-bit 定義，在運算時直接轉為 32-bit，所以我們有機會透過 SIMD(Single Instruction Multiple Data)的概念，用一個指令獲取 4 個 8-bit，提升四倍的平行度

為了一次讀取 4 個 8-bit 指令，雖然可以使用 32-bit，但運算中的溢位(overflow)問題則需要額外解決。為求實驗方便，我們直接將 8-bit sign extension 到 32-bit，並定義了新的暫存器 V(4x32-bit)來讀取 4 個 32-bit 指令，結合 trv32p3x 原本的暫存器 X(32-bit)，進行擴展指令集設計。

表二、RISC-V 擴展指令集

RISC-V 擴展指令集		說明
vsplit	vd(V), rs1(X)	將 rs1 拆分、延展(sign extension)成 vd
vmul_of	vd(V), vs1(V), vs2(V)	將 vs1 和 vs2 相乘，存在 vd
vsum	rd(X), rs1(X), vs2(V)	將 rs1 和 vs2 中四段 8-bit 進行累加
vmacc_of	rd(X), vs1(V), vs2(V)	結合兩個指令：vmul_of 和 vsum
nvmacc_of	rd(X), rs2(X), rs3(X)	最上層指令，結合 vsplit 和 vmacc_of



圖三、nvmacc_of 指令行為

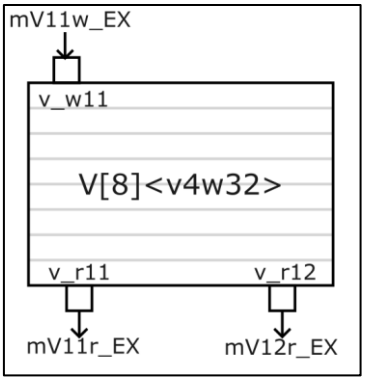
將 rs2 和 rs3 以 vsplit 拆分成 4 個 8-bit，再用 sign extension 延展成 4 個 32-bit 存入 V 暫存器。vmul_of 會將 rs2 加上 offset rof 後，每 32-bit 逐段和 rs3 相

乘，並逐段存入 vsum 的其中一個 operand，vsum 再讓 rs1 和 vmul_of 的 output 逐段相乘累加，並存入 rd(暫存器 X)裡。因為是進行 MAC 的運算，所以 rd 其實也是 rs1。

2-2. RISC-V 處理器架構設計

根據所定義的擴展指令集，我們在硬體方面也要做出相應的調整。

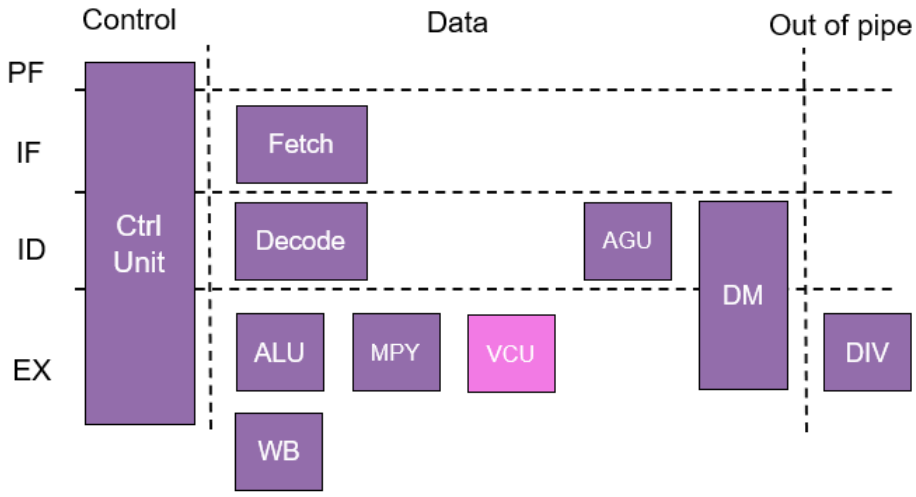
2-2-1. 向量暫存器設計



圖四、向量暫存器 V 架構

定義一個新的暫存器 V，共有 8 個向量，每個向量是 4x32-bit。共有一個寫入 port(4x32-bit)和兩個讀取 port(4x32-bit)。

2-2-2. VCU(Vector Compute Unit)設計



圖五、新增 VCU 的 trv32p3x 架構

我們將擴展指令集的功能整合，定義 I/O 與相互的連結，形成新的功能單元 VCU，並與 trv32p3x 進行整合。由圖可知，VCU 僅在 EX stage 產生行為。

2.5 軟體方面的調整

在定義了擴展指令集和硬體架構後，我們須在軟體方面做出相應調整，以將我們設計的指令融合在軟體中。首先，我們確認 convolution 剩餘的深度(filter input depth)不小於 4，以確定可以進行 SIMD，否則使用原本的指令。再來，我們以 nvmacc_of 取代原本的 MAC 運算。最後，迴圈執行次數變為原本的四分之一(右移兩 bit)。

3. 實驗結果

將修改過，使用擴展指令集的 trv32p3x_modified 和 trv32p3x，使用 ChessDE 對 MLperf Tiny 的四種 AI 模型進行指令集模擬，得到週期數和指令數等結果。須特別注意的是，在 ChessDE 指令集模擬中，工具會摻入 debug code 以利除錯，而此功能會大量增加週期數和指令數。透過添加參數 NDEBUG 即可不使用此功能，模擬結果會更貼近實際結果。

3-1. Convolution 最內層迴圈的模擬結果

我們對比修改後和修改前的結果，修改前共 4 個週期，有 2 個 lb、兩個 add、和一個 mul，因為兩個 lb 的位址連續，工具自動使用 post increment 和 ILP 並行指令，可以減少一個週期。修改過後，共 3 個週期，有 2 個 lw 和一個 nvmacc_of。可見經過特殊指令集設計，我們將 MAC 運算的週期從 4 降為 3。

3-2. ConvPerChannel 最內層迴圈週期數

表三、ConvPerChannel 最內層迴圈週期數模擬結果

ConvPerChannel Innermost Loop Cycle Count	Original	Modified	Cycle count reduction
Keyword Spotting	2,318,720	512,000	78%
Visual Wake Words	6,683,928	1,548,288	77%

Image Classification	11,573,952	2,787,456	76%
Anomaly Detection	0	0	
Sum	20, 576,600	4,847,744	76%

對 ConvPerChannel 最內層迴圈的週期數進行分析，可以最直觀的觀察 MAC 運算的效能提升。修改過後的 ConvPerChennel 有著四倍的平行度，以及最內層迴圈從 4 個週期變為 3 個週期，理論上週期數會減少 13/16(81.3%)，然而 ConvPerChannel 的深度並非全部大於 4，對於深度小於 4 的情況則不能使用 SIMD，所以實驗結果大概在減少 76%左右。由於 Anomaly Detection 並未使用 ConvPerChannel 運算，所以有關此運算的統計皆是 0。

3-3. ConvPerChannel 模擬結果

表四、ConvPerChannel 週期數模擬結果

ConvPerChannel Cycle Count	Original	Modified	Cycle count reduction
Keyword Spotting	15,369,365	9,605,536	37%
Visual Wake Words	40,585,506	20,907,079	49%
Image Classification	59,008,360	23,942,139	59%
Anomaly Detection	0	0	
Sum	114,963,231	54,454,754	52%

我們現在轉而觀察 ConvPerChannel 整體的週期數，由於 ConvPerChannel 除了最內層的 MAC 運算以外，還有其它的運算並未被加速到，所以週期數的減少比例有所下降，對四個模型加總取平均後可知，減少大概在 52%左右。

3-4. 整體週期數模擬結果

表五、整體週期數模擬結果

Total Cycle Count	Original	Modified	Cycle count reduction
Keyword Spotting	25,696,235	19,932,406	22%
Visual Wake Words	71,655,446	51,977,019	27%
Image Classification	69,081,085	34,014,864	51%

Anomaly Detection	1,985,939	1,985,939	0%
Sum	168,418,705	107,910,228	36%

我們統計模擬結果的整體週期數，由於不同模型中 ConvPerChannel 運算的佔比不同(如圖三所示)，加速 ConvPerChannel 對於整體週期數的影響比例也因此不同。而 Anomaly Detection 並未使用 ConvPerChannel，所以並未被加速到。將四個 AI 應用加總取平均，我們得知修改過的 trv32p3x 對於 MLperf Tiny 有著 36%的提升。

3-5. 指令數模擬結果

除了週期數的統計之外，我們也進行指令數的統計。但兩者在減少的比例上有著類似的結果，所以我們就省略有關指令數模擬結果的探討。

4. 結論

在結論中，我們將討論實驗結果和研究未來的延伸方向。

4-1. 實驗結果

我們利用了 SIMD、special instruction 等概念進行擴展指令集設計，最後在 MLperf Tiny 基準上達成平均 36%的週期數減少。其中，在 ConvPerChannel 最內層迴圈達到平均 76%的週期數減少，距離理論上限 81.3%有著 5%的差距，在考量 filter input depth 不總是小於 4 的情況後，算符合實驗預期。

4-2. 研究延伸方向

由於時間因素，我們的研究並未探討完此設計的所有可能性，在未來我們還有許多改進方向。

4-2-1. 其它 AI 運算的優化

我們只對佔比最大的運算 ConvPerChannel 進行優化，而佔比第二、第三大的 FullyConnected、DepthwiseConvPerChannel 仍可以繼續進行優化，其中，

DepthwiseConvPerChannel 和 ConvPerChannel 有著更高的相似性，可以是未來優化的主要方向。

4-2-2. V 暫存器和 X 暫存器的整合

為了實現 SIMD 指令，我們需一次讀取 4 個 8-bit 資料，為了處理溢位問題，我們直接選擇將 8-bit sign extension 至 32-bit，並定義了新的暫存器 V 來進行運算。我們雖然因此學會了自定義暫存器並與指令集整合，但我們未必需要定義新的暫存器。我們可以透過設計妥善處理溢位問題，並將其整合到原本的 X 暫存器上，否則定義新的 V 暫存器會大幅的增加面積而得不償失。

4-2-3. 考量 PPA 的設計

在本研究中，我們尚未將 PPA(Performance, Power, Area)納入考量，僅探討指令層級的可行性。未來我們可以透過工具將 nML 轉換成 RTL，並在 FPGA 上進行合成與驗證，並探討不同設計方式對 PPA 的影響。

5. 參考資料

[1] M. Ali and D. Göhringer, "Application Specific Instruction-Set Processors for Machine Learning Applications," 2022 International Conference on Field-Programmable Technology (ICFPT), Hong Kong, 2022, pp. 1-4.

[2] Synopsys, Inc., “ASIP Designer Tinycore2 Workshop Lab Guide V-2024.06”, 2024.

[3] Synopsys, Inc., “ASIP Designer Trv (RISC-V ISA) Models Processor Manual V-2024.06”, 2024.

[4] Synopsys, Inc., “ASIP Designer Trv (RISC-V ISA) Simple Data path Extensions (SDX) Processor Manual V-2024.06”, 2024.

[5] Werner Geurts., “RISC-V with SIMD Extensions for Matrix Multiplication”, 2023.

6. 心得感想

本研究協同 Synopsys ASIP Designer Contest 同時進行，在此之前，我們花了數個月進行 FPGA 硬體加速器相關的訓練。所以在接到這個比賽的題目時，我們已經有了部分處理器設計的知識，而 ASIP Designer 所提供的工具和設計流程，有別於傳統的 RTL 設計，可以讓處理器設計流程更快的佚代，給我們不同的設計觀點。透過這個專題，我們學習到了不少有關於 RISC-V 處理器的設計思維及架構，並能夠依此創造出專用的優化設計。無論是資料蒐集、電路設計、報告撰寫及成果發表都使我受益良多。因為比賽時間短暫，且需要熟悉全新的工具和設計流程，我們經歷了許多困難。但透過團隊合作和指導教授的指導，我們還是克服了困難完成設計。