12/13 Ultra Lightweight Dehaze Methods for Robot Vision Using High-Level Synthesis

國立臺北大學 電机系 宋啟嘉 特聘教授

- FPGA Trends

△先進製程與規模：目前最先進技術採用 7nm FinFET+ 製程與 300 mm 晶圓（如 Xilinx Versal平台）主要針對雲端中心的 FPGA 加速卡

△ Virtex UltraScale+ VU19P：採用 16 nm 製程，擁有 350 億個電晶体，由 4 個晶片透過中介層（Interposer）連接，擁有約 900 萬個邏輯單元，是目前世界上最大的 FPGA

△成本效應：邏輯閘成本大幅下降：1990 年的 1 美元/gate 降至 2020 年的 0.01 美元/gate

△雲端應用：AWS 自 2018 年起採用 FPGA 實例，目前佔比已超過 50%

△異質硬体加速：整合了 ARM CPU 與高頻寬 ADC/DAC，適用於多種用途，如 ADAS 和 SDR

- FPGA 架構演進史

1. FPGA（Since 1980）：
   - 架構：主要由 Logic blocks 組成，含少量 Block RAM 和 PLL
   - 特点：使用 HDL（Verilog / VHDL）進行硬体加速，並行處理能力強，但編程靈活性低
2. SoPC FPGA（2004 - 2012）：
   - 架構：傳統 FPGA 配置 Soft Core（如 Nios，MicroBlaze）
   - 特点：結合硬体加速与 C 語言編程，實現軟体協同設計，但速度受限於軟核 CPU 的效能
3. SoC FPGA（2012 - 2017）：
   - 架構：整合實体 32 - bit CPU Core（ARM Cortex A9 / A8）与 FPGA 架構
   - 特点：CPU 速度（677 MHz - 1 GHz）不再受 FPGA 邏輯速度限制，且高於 FPGA，Linux 嵌入式系統整合更容易
4. Cloud FPGA（2019 - Today）：
   - 架構：針对伺服器市場的虛擬實例
   - 特点：能源效率優於 CPU / GPU，支援使用 C/C++ or OpenCL 撰寫加速代碼（如 Intel DevCloud
5. MPSoC FPGA（2017 - Today）：
   - 架構：多核心 SoC FPGA，整合 64-bit ARM Cores，GPU 与即時處理器
   - 特点：支持視覺處理，獨立的 RTPU 允許系統進入低功耗休眠模式
6. RF FPGA / RFSoC（2017 - Today）
   - 架構：整合 RF-sampling 数據轉換器（ADC / DAC）的適應性平台
   - 特点：取代外部轉換器，減少 50% 功耗与佔用空間，將大部分 RF 訊号處理移至数位領域

- Multiple Core FPGA

核心概念：多核心處理器 + FPGA 用於 AI、机器學習、模式識別、ADAS 和 SDR

廠商对比：△ MPSoC ： Xilinx + ARM v.s Intel + Altera (Xeon Phi)
△ RFSoC ： Xilinx + ARM + ADC v.s Intel RF FPGA

雲端範例 (Data Center Acceleration)：
△ 應用於影像識別、資安加密、大数據压縮
△ 透过整合 (CPU + FPGA)，可降低總体擁有成本 (TCO)並提升 2倍效能

- Heterogeneous Computing FPGA

核心觀点：異質 SoC FPGA 潛力巨大，但要釋放其全部效能需要極其廣泛的專業知識

技能需求層級：1. 基礎層：基頻通訊設計、基板支援包設計
2. 軟体層：協定軟体設計、嵌入式軟体設計.
3. 類比 / 射頻層：類比系統設計、射頻系統設計
4. 頂層：整合前最佳化 —— 這是最關鍵但也最具挑戰性的部分

- 優化技術 - Optimization 1: Loop Unrolling

利用迴圈迭代的平行性

運作方式：實例化更多硬体資源，讓多個計算同時執行，而非排隊執行

- HLS 實戰結果 - QRD 雷達設計

關鍵時間：HLS 展現巨大優勢，從 12週縮短到 1週

效能：HLS 版本 (21ms) 優於手寫版本 (37ms)

資源消耗：HLS 在記憶体使用上顯著較少，暫存器和邏輯元的使用量也更低

結論，使用HLS 能節省大量工程時間，且不犧牲效能

- HLS 實戰成果 - 霧氣務隊系統 (DCP Fog)

不同配置 (Profiles)：比較 Profile 1 至 Profile 6 的效能差異

關鍵数據：介面寬度：從 64-bit 提升至 256-bit
FPS (幀率)：Profile 1 僅 10 FPS，優化後的 Profile 6 高達 529 FPS
代價：隨著效能提升，LUT 與 FF 的資源使用率也顯之增加

能效比：Profile 6 雖然功耗較高，但能效表現最佳

- High-Level Synthesis，HLS (高階合成技術)

△ 定義：將高階語言 (C/C++ / System C) 編譯轉換为 RTL 电路

△ 優勢：降低硬体設計門檻
降低設計成本

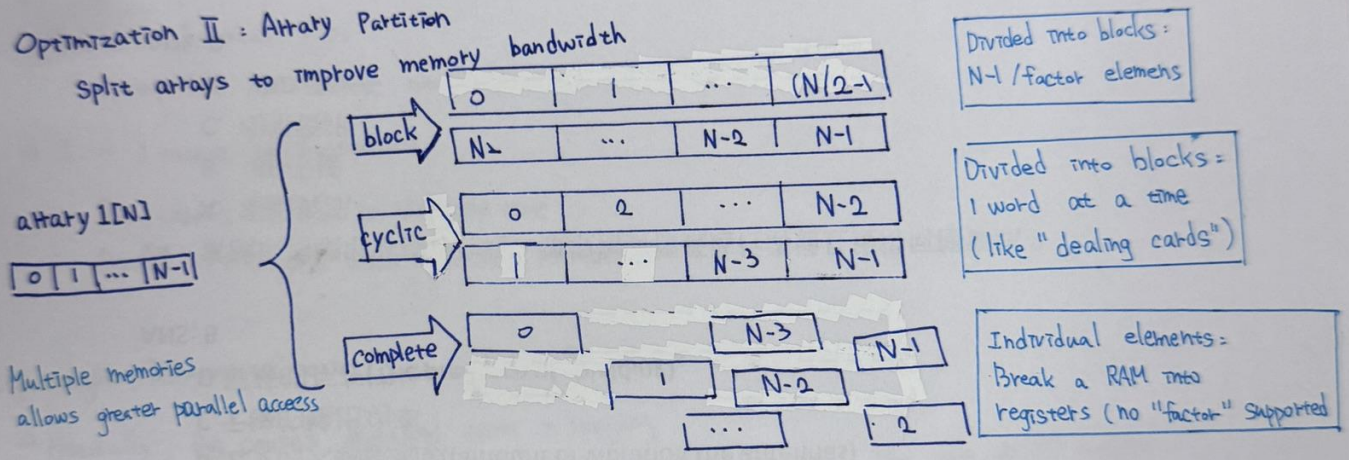
△ 流程：輸入 C Code 和 Directives → Vivado HLS 進行控制和資料流萃取 → 輸出 Verilog /VHDL實作

△ 靈活性：透过不同的 Directives，同一份程式碼可產生多種不同的硬体實作 (如面積最小化 or 速度最佳化)

- PYNQ Heterogeneous Hardware Acceleration
  - △ 名稱含義：Python + ZYNQ
  - △ 目標：利用 Python 的高生產力來開發 Zynq 嵌入式系統
  - △ 架構堆疊（Stack）：應用層：使用 Jupyter Notebooks，Python 函式庫
    API 后端动態：負責軟体与硬体溝通
    硬体層（Overlay）：FPGA 由 Bitstream，可視为可載入的硬体函式庫
  - △ 優点：讓不熟悉底層硬体描述語言的軟体工程師，也能輕鬆調用 FPGA 加速功能

- Optimization I：Loop Unrolling
  - △ 利用空間換取時間來開發迴圈的平行性
  - △ 運作原理：預設（Standard HLS）：迴圈折疊，硬体資源共用，依序執行。耗時 4 Cycles
    展開（Unrolling）：複制多份運算元，同時執行計算。耗時 1 Cycle
  - △ 程式碼：for(i=3; i>=0; i--) 通过展開指令，讓硬体一次做完 4 次加法

- Optimization II：Attary Partition
  Split arrays to improve memory bandwidth

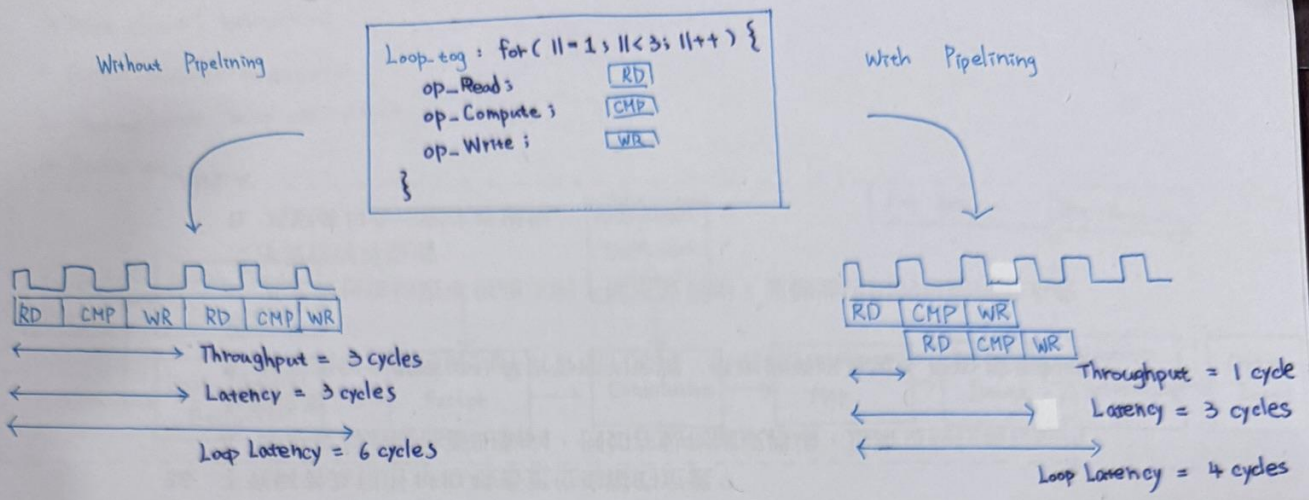


- Optimization III：Resources
  - △ Allocation directive constrains resources
    - Operations
      Number of adders instantiated RTL
      Can save a lot of area
  - △ Specify implementation
    - Tag operator
    - Select core for this Mult

```
thisMult = b[i] * c[i];
a[i] = thisMul
```

Core	Description
Mul	Combinational mult
Mul3s	3-Stage pipelined mult
MulnS	HLS determine stages

- Optimization IV : Loop Pipelining

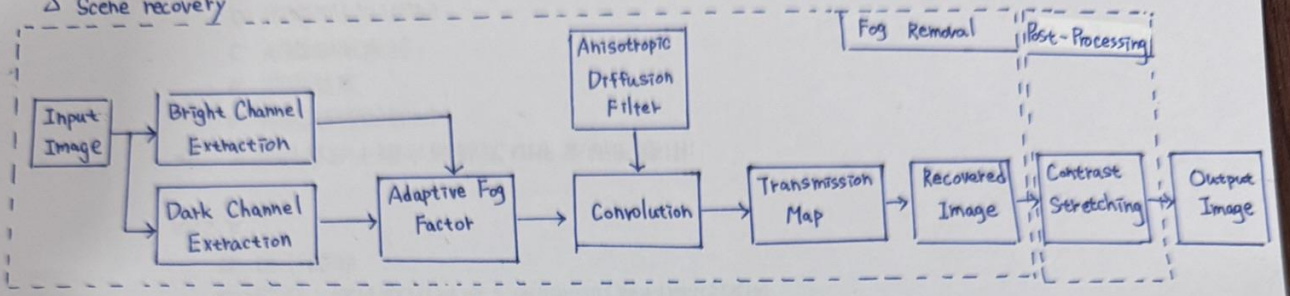


- Optimization IV : Loop Pipelining
  - △ Iteration Interval (II)
    Cycles loop must before next iteration
  - △ II = 1 cannot be implemented
    Port cannot be read at the same time
    Similar effect with other resource limitations

- Lightweight Dehazer Circuit System Design for Robot Vision
  - △ Naturally, fog causes degraded vision in imaging
  - △ Degrades performance of Robot Vision using Machine Learning, like yolo or ViT models
    - Object detection
    - Classification
    - Tracking
    - Segemetation

- Main Objective
  - △ Develop a ultra dehazing (defog, derain, low light, under water) algorithim that performs better or on-par with current works
  - △ Eliminate or minimze the drawbacks of existing method
  - △ Minimizing computation requirements
  - △ Implementation in FPGA as an IP using HLS
  - △ Integrate into HDMI pipelining

- Dehaze Method
  - △ Dark channel extraction
  - △ Bright channel extraction
  - △ Transmission map estimation
  - △ Scene recovery



- Dehazing Results
  - △ Eliminates color shifting from original DCP
  - △ Preserves image naturalness

- Dehazing Metrics on RESIDE
  - △ Tested on MATLAB R2024b using CPU-only process
  - △ High PSNR and SSIM compared to conventional method
  - △ Higher throughput compared to other methods

- Dehazing Result on Computer Vision
  - △ Detection test using YOLO v7-tiny pretrained model
  - △ Up to 7.9% mAP improvement on very dense haze

- FPGA Video Pipelining
  - △ Ping-Pong buffer implementation
  - △ 2 frames latency on 1920 x 1080 30FPS
  - △ Adjustable tuning parameters on-the-fly

- Resources Utilization
  - △ Utilization reports from Vitis and Vivado toolchain
  - △ Minimum resources utilization footprint

- Undergoing Work:
  - △ Run segmentation, detection, and classification with YOLO v7, v12
  - △ Object Detection: Locate and classify multiple objects
  - △ Object Segmentation: Provide pixel-level masks for instance/semantic segmentation
  - △ Image Classification: Assign a global label to an image (scene or object category)