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

---

日期：2025/12/23

講者： 宋啟嘉

FPGA 發展趨勢

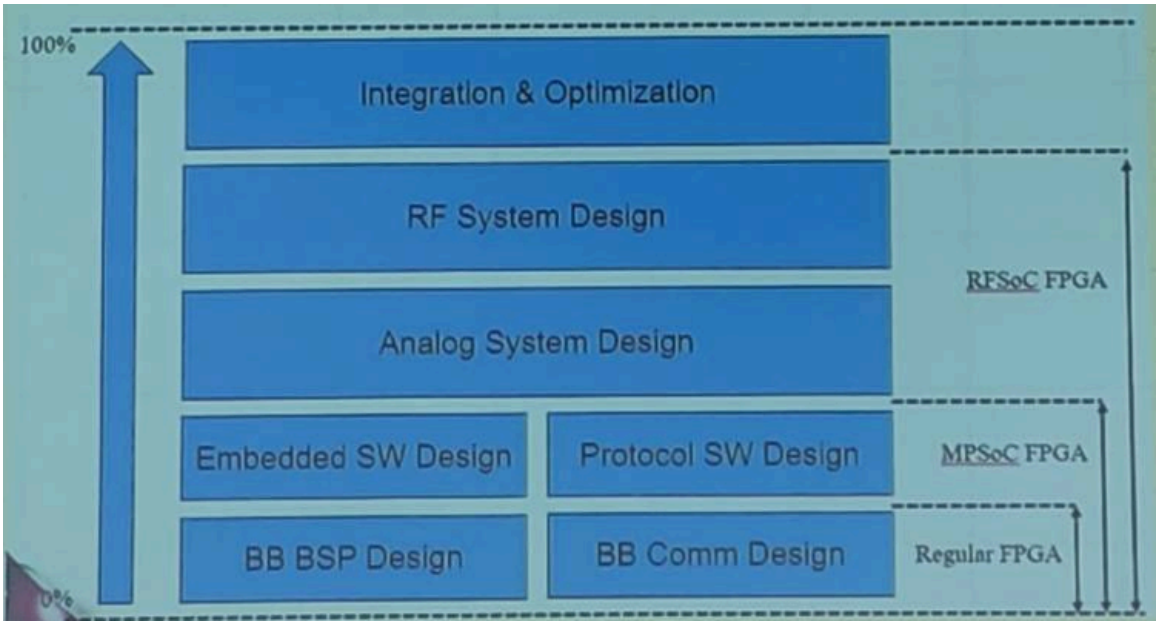
- 製程：**7nm FinFET+**
- 成本：**1990：$1 → 2020：$0.01**
- 雲端：AWS 自 2018 起大量導入 FPGA（>50%）
- 架構趨勢：**異質加速（ARM + FPGA + ADC/DAC）**

FPGA 架構演進脈絡

FPGA → SOPC → SoC FPGA → MPSoC → Cloud FPGA → RFSoC

- **FPGA（~1980）**：純邏輯，HDL 開發
- **SOPC FPGA（2004–2012）**：Soft CPU（Nios / MicroBlaze）
- **SoC FPGA（2012–2017）**：整合 ARM Cortex-A + Linux
- **MPSoC FPGA（2017–至今）**：多核心 ARM / GPU / RT core
- **Cloud FPGA（2019–至今）**：資料中心虛擬化 FPGA
- **RFSoC FPGA（2017–至今）**：ADC/DAC + RF 直接整合

異質運算 FPGA



多核心 FPGA

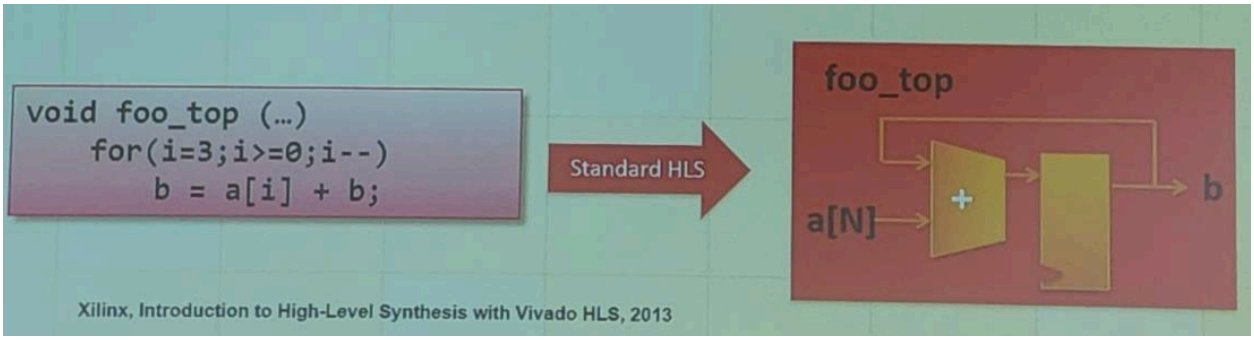
- 組合：**Multi-core CPU + FPGA**
- 應用：AI、ML、Pattern Recognition、ADAS、SDR
- 平台對比：
  - Xilinx + ARM → MPSoC / RFSoC
  - Intel + Altera → Xeon Phi / RF FPGA

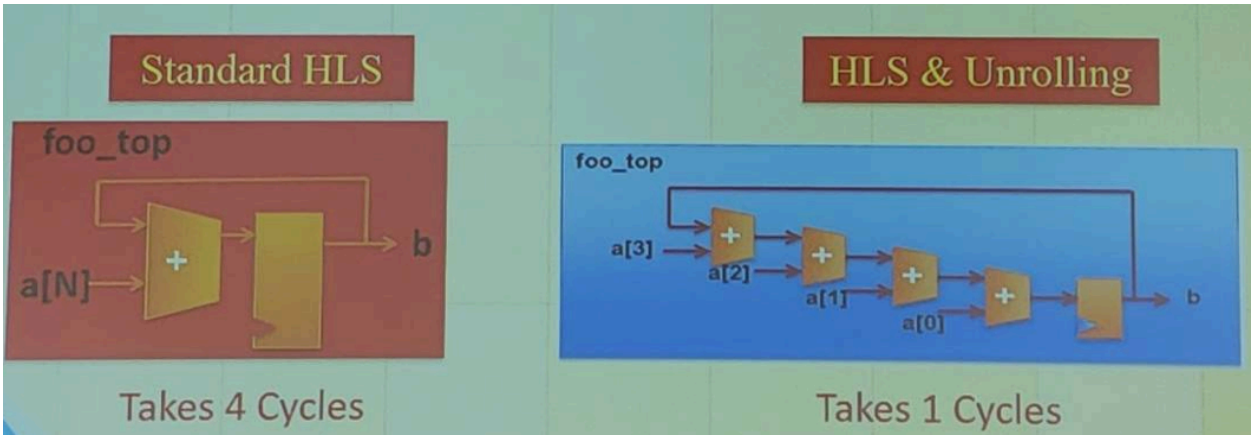
HLS（High-Level Synthesis）

- **C/C++ → RTL**
- 目的：
  - 降低硬體開發門檻
  - 加速設計探索
- 特點：
  - 可從同一份 C code 產生多種 RTL
  - 透過 directives 控制效能 / 面積

Optimization

Loop Unrolling

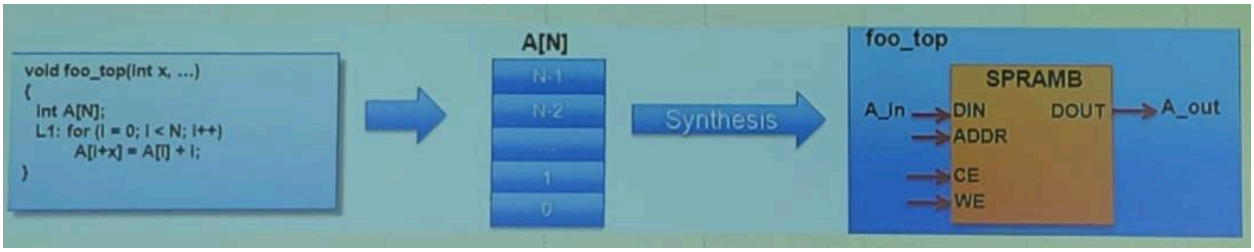


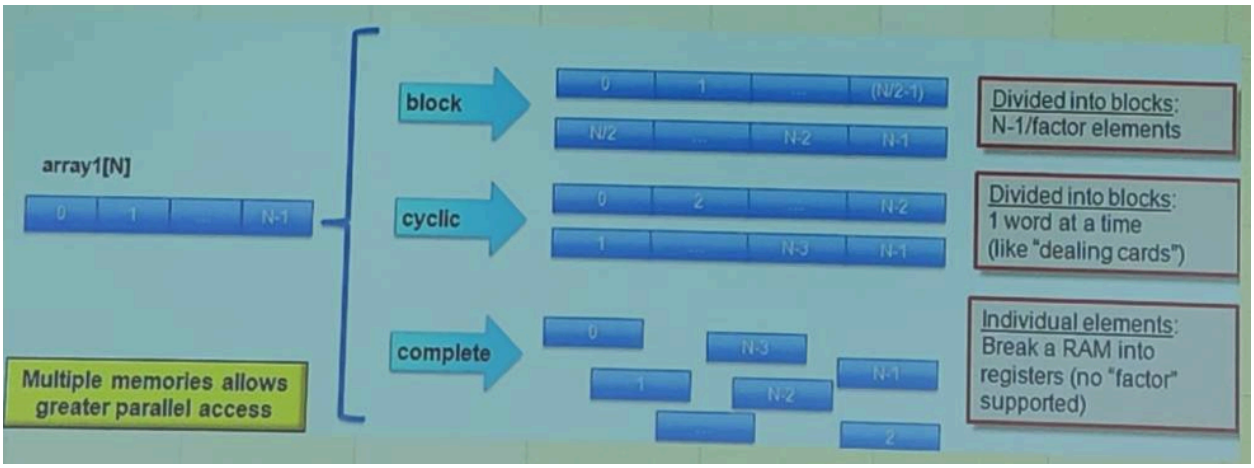


- 預設：loop 是 **rolled**（一個加法器重複用）
- Unrolling：
  - 複製硬體資源
  - 多個 iteration 同時執行
- 效果：
  - Latency ↓（例如 4 cycles → 1 cycle）
  - Area ↑（硬體用量增加）

> 💡 **提升平行度（Parallelism），用面積換速度**

Array Partitioning

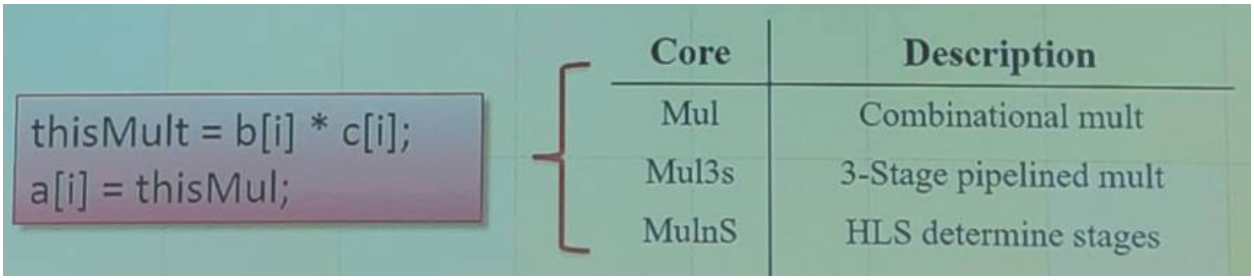




- HLS 預設：array → RAM（一次只能存取有限筆）
- Partition 方式：
  - **block**：連續切塊
  - **cyclic**：交錯分配（像發牌）
  - **complete**：每個 element 一個 register
- 效果：
  - 允許多筆資料同時讀寫
  - 支援 unrolling / pipelining

> 💡 **提高記憶體頻寬，不 partition，unroll 也跑不快**

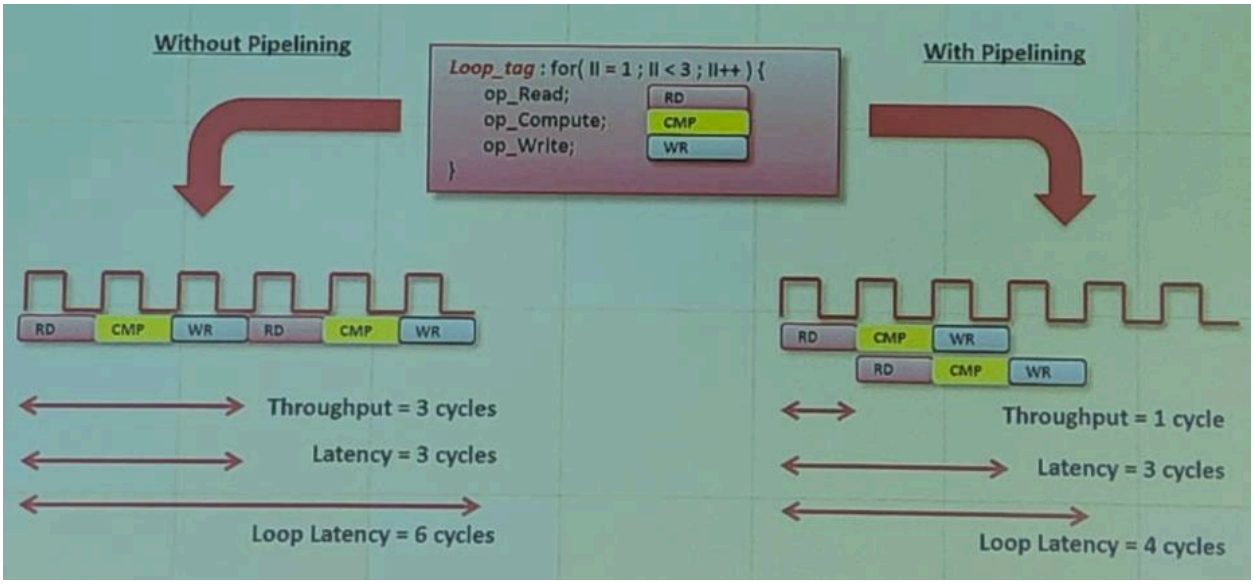
Resource Allocation

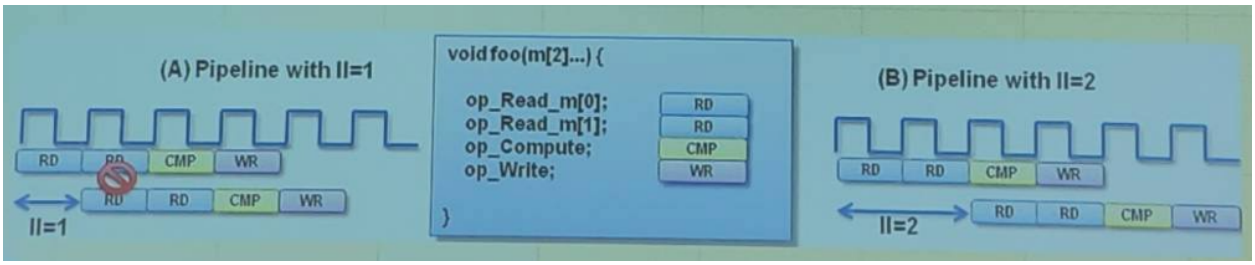


- 指定運算資源數量（如 adder / multiplier）
- 可指定 core 類型：
  - Mul：組合邏輯
  - Mul3s：3-stage pipeline
  - MulnS：HLS 自動決定
- 效果：
  - Area ↓
  - Latency 可能 ↑

> 💡 **控制面積，用 directive 控制硬體「長怎樣」**

Loop Pipelining





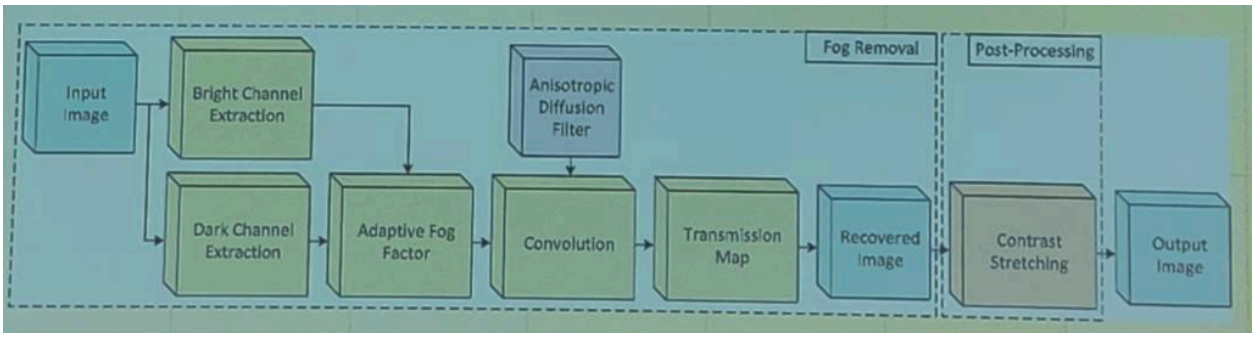
- 核心概念：**Iteration Interval (II)**
  - II = 每隔幾個 cycle 開始下一次 iteration
- 理想：II = 1（每 cycle 一筆資料）
- 但 II=1 可能失敗原因：
  - Memory port 不夠
  - 資源衝突
- 解法：
  - Array partition
  - 放寬 II（例如 II=2）

> 💡 **提高 Throughput**

Dehazer

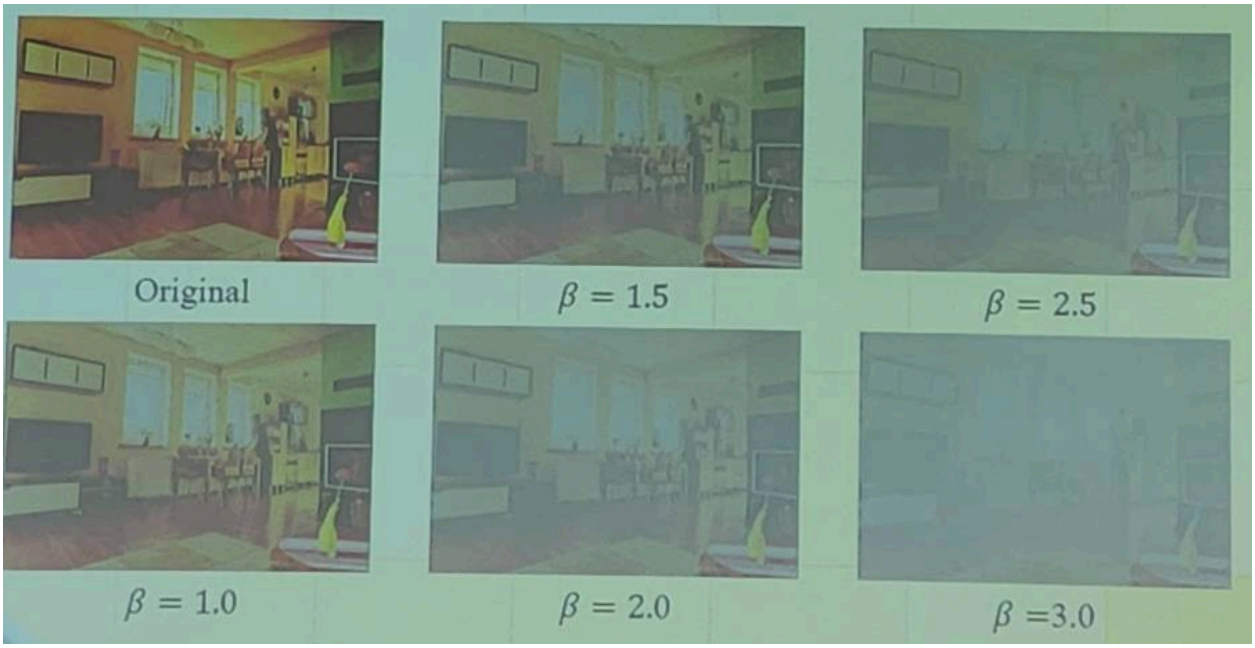
- 霧氣會影響：
  - Object detection
  - Classification
  - Tracking
  - Segmentation
- 對 YOLO / ViT 等模型效能有負面影響
- **前處理（Dehaze）可以幫助後端 AI**

Dehaze 方法流程

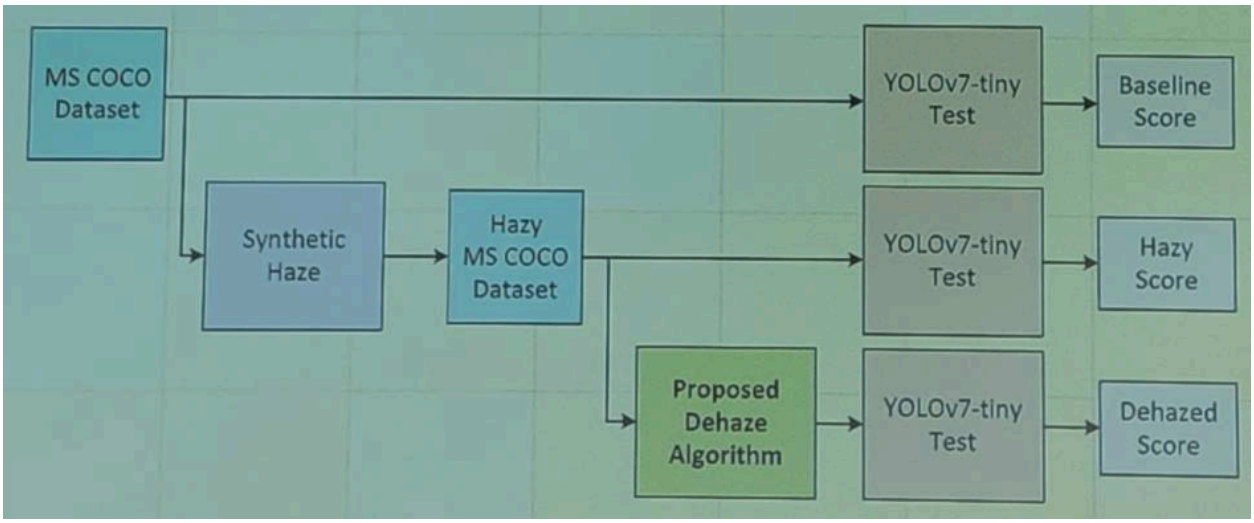


Synthetic Haze（COCO Dataset）

- 使用 **MiDaS depth estimator** 產生 depth map
- 基於 **atmospheric scattering model** 合成霧氣
- 霧濃度以 **β = 1.0 ~ 3.0** 控制（越大越濃）
- 建立 **多等級霧氣 COCO dataset**



Computer Vision 評估流程



- Dataset：MS COCO
- Model：**YOLOv7-tiny**
- 比較三組：
  1. Baseline（原始 COCO）
  2. Hazy COCO
  3. Dehazed COCO（Proposed）
