

Edge Computing with DSP Acceleration

Lab 1

Department of Electronic and Computer Engineering
National Taiwan University of Science and Technology

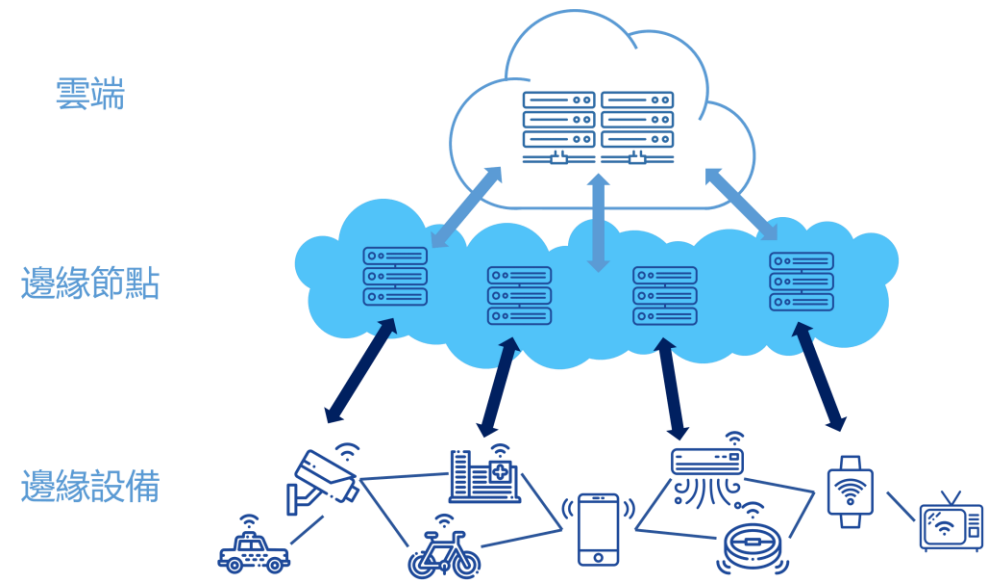
What is Edge Computing?

■ Distributed Computing Architecture

- processing and analyzing data closer to the data source
- enhance data processing efficiency, improve real-time capabilities.

■ Characteristics

- Real-time and Low Latency
- Bandwidth Efficiency
- Privacy and Security
- Cloud Collaboration



TDA4VM重點功能特色

- ❖ 嵌入式系統即時運算
- ❖ 為邊緣運算、自動駕駛應用設計
- ❖ 雙核心Arm® Cortex®-A72
- ❖ 整合了三個DSP
 - 提供矩陣運算加速
 - 可應用在深度學習、影像辨識



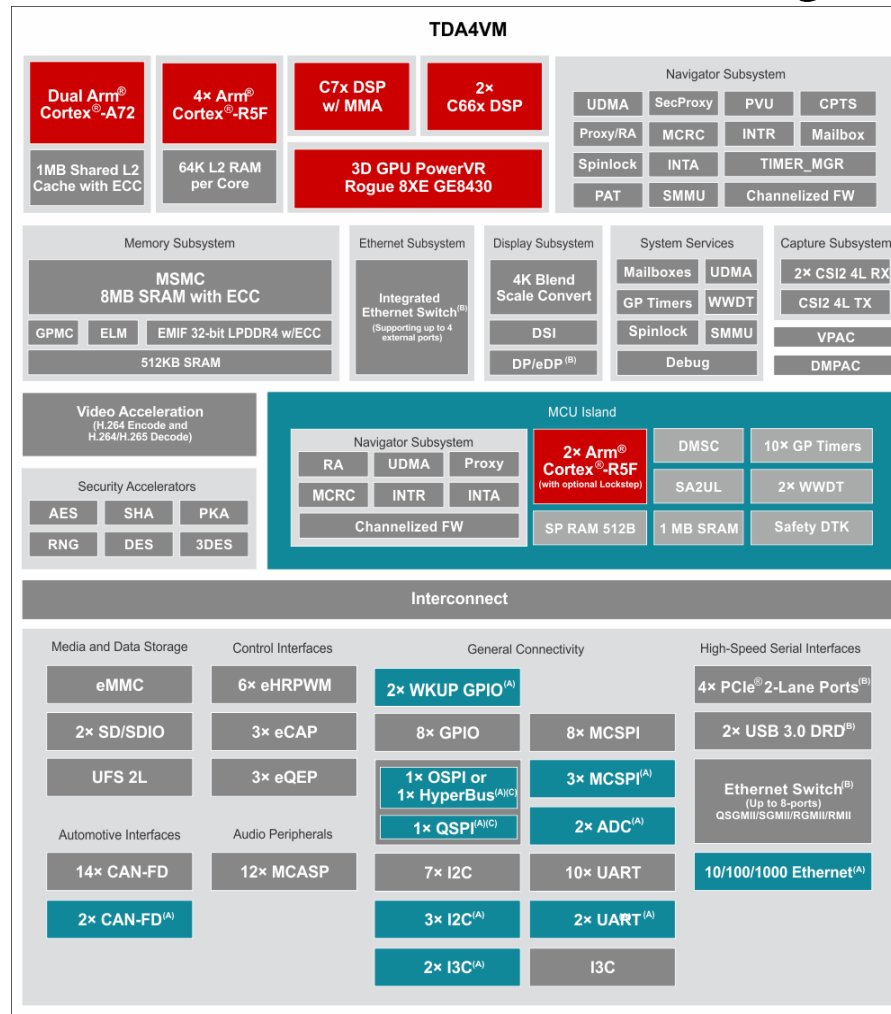
TDA4VM重點功能特色

- ❖ C71x DSP MMA提供了8TOPs的矩陣加速運算性能和 80FLOPs 浮點運算性能，還有C66x DSP 加持
- ❖ 雖然算力不高，但其能在15瓦不到和0.1公斤的小型開發板上進行影像辨識，甚至不用風扇散熱

品項	CPU	Memory	AI運算單元	效能	重量
Nvidia Drive AGX Xavier	8 Core Carmel ARM64	16 GB DDR4	Nvidia Volta 512 core	30 TOPS	1.5公斤
Nvidia Drive Orin	12 Core ARM Cortex A78AE	32 GB DDR5	2x Nvidia Ampere	400 TOPS	1.6公斤
德州儀器 TDA4VMXEVN	2 Arm Cortex-A72	4 GB DDR4	C7x DSP with MMA (Matrix multiply accelerator)	8 TOPS	0.1 公斤

SK-TDA4VM Embedded Environment Platform

■ TDA4VM Architecture Diagram



SDK:

<https://www.ti.com/tool/download/PROCESSOR-SDK-LINUX-SK-TDA4VM>

Fetching software:

<https://www.balena.io/etcher>

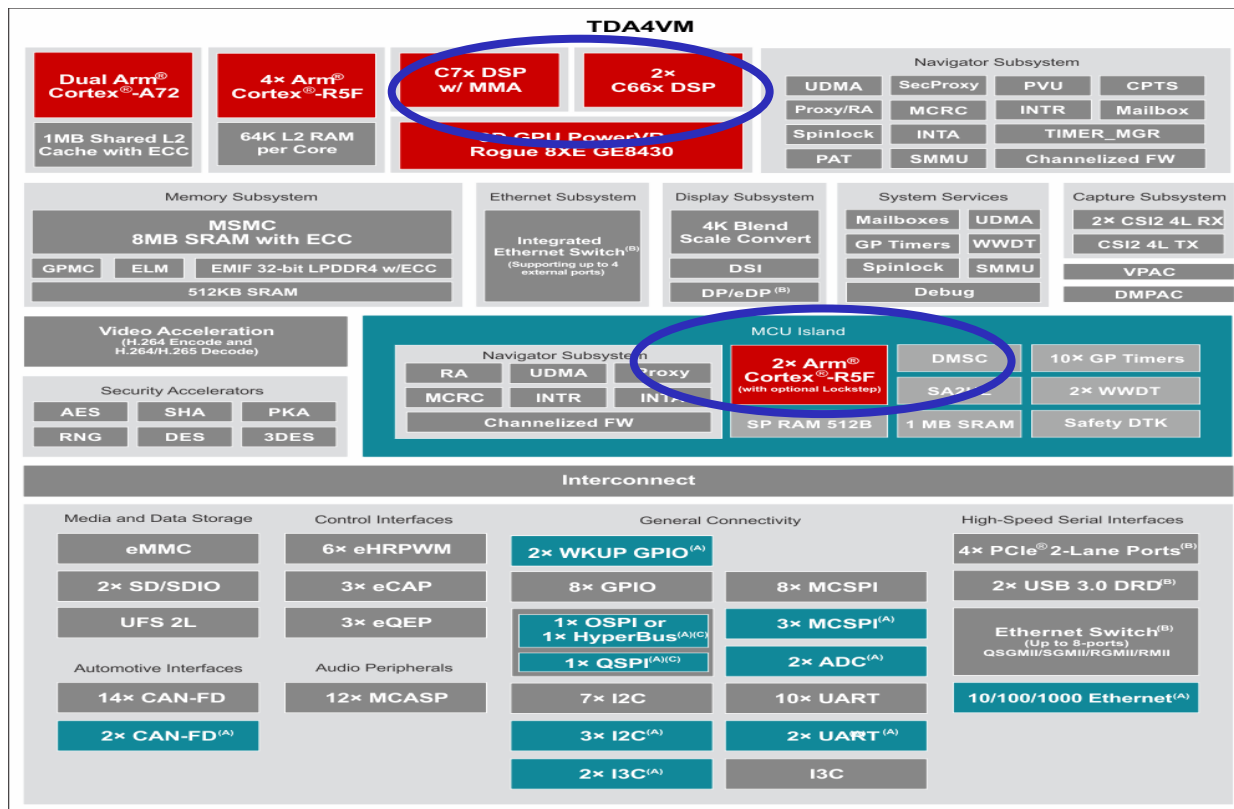
Dual Arm Cortex A72:多核處理器，具有高效能和低功耗的特點。

C7x DSP with MMA :搭載了矩陣乘法加速器 (Matrix multiply accelerator, MMA) 來針對深度神經網路進行運算。

C66x DSP:浮點數的向量運算 (Vector Processing) 以及 SIMD 運算

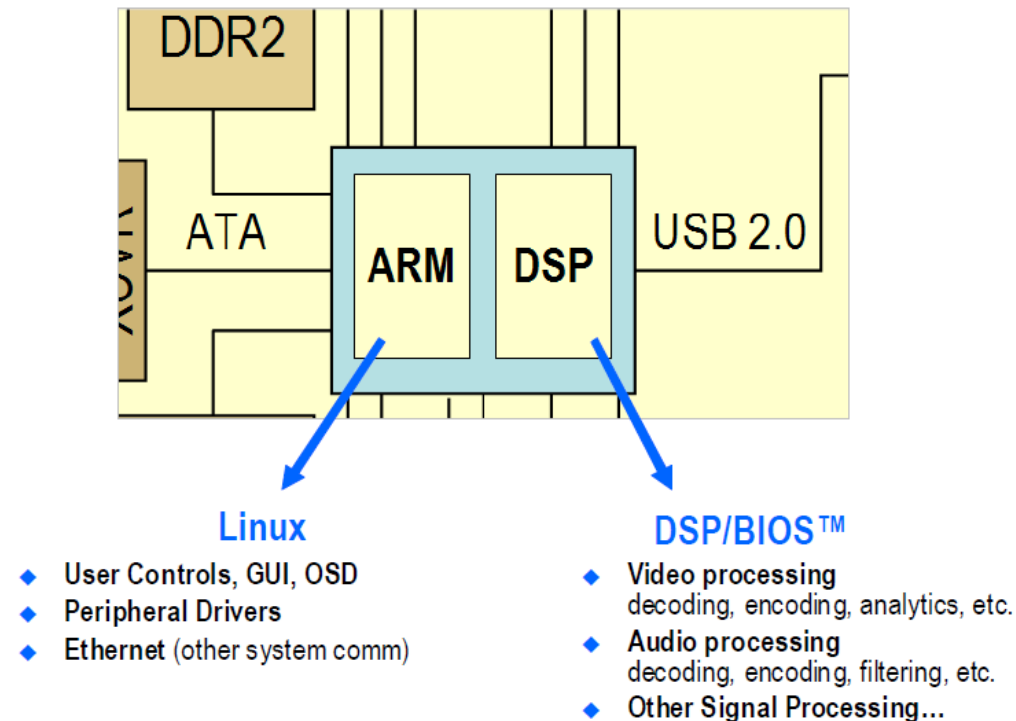
Software Design based on Embedded Processors

- ❖ Developing pure software vs. developing embedded software: **Understanding your hardware.**



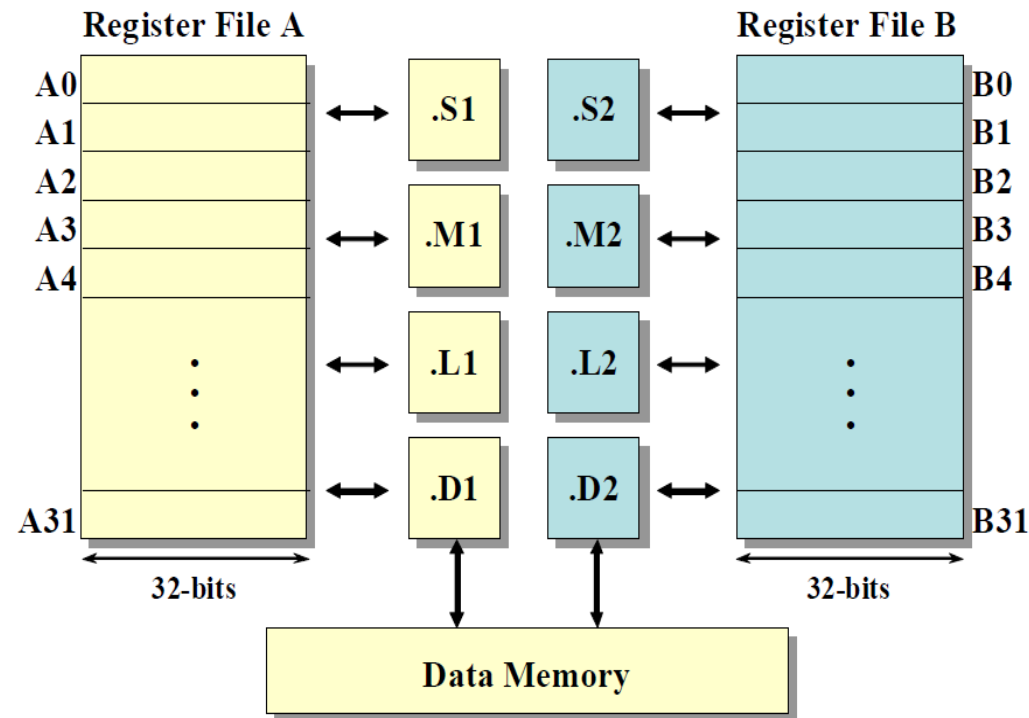
Software Design based on Embedded Processors

- ❖ Different types of processors play different roles in edge computing platforms.



Software Design based on Embedded Processors

❖ A typical hardware architecture of the DSP.



$$Y = \sum_{n=1}^{40} a_n * x_n$$

```

MVK .S1    40, A2      ; A2 = 40, loop count
loop: LDH   .D1    *A5++, A0  ; A0 = a(n)
      LDH   .D1    *A6++, A1  ; A1 = x(n)
      MPY   .M1    A0, A1, A3  ; A3 = a(n) * x(n)
      ADD   .L1    A3, A4, A4  ; Y = Y + A3
      SUB   .L1    A2, 1, A2   ; decrement loop count
[A2] B     .S1     loop      ; if A2 ≠ 0, branch
      STH   .D1    A4, *A7     ; *A7 = Y
  
```


Software Design based on Embedded Processors

- ❖ An algorithm-level consideration of performance.
- ❖ **Goal:** Maximizing the utilization of function units.

Serial	Partially Parallel	Fully Parallel
B .S1	B .S1	B .S1
MVK .S1	MVK .S2	MVK .S2
ADD .L1	ADD .L1	ADD .L1
ADD .L1	ADD .L2	ADD .L2
MPY .M1	MPY .M1	MPY .M1
MPY .M1	MPY .M1	MPY .M2
LDW .D1	LDW .D1	LDW .D1
LDB .D1	LDB .D2	LDB .D2

```

loop:
    ldh.d1  *A8++,A2
    ldh.d1  *A9++,A3
    nop     4

    mpy.m1  A2,A3,A4
    nop
    add.l1  A4,A6,A6

    sub.l2  B0,1,B0
[b0] b.s1      loop
    nop     5
  
```



```

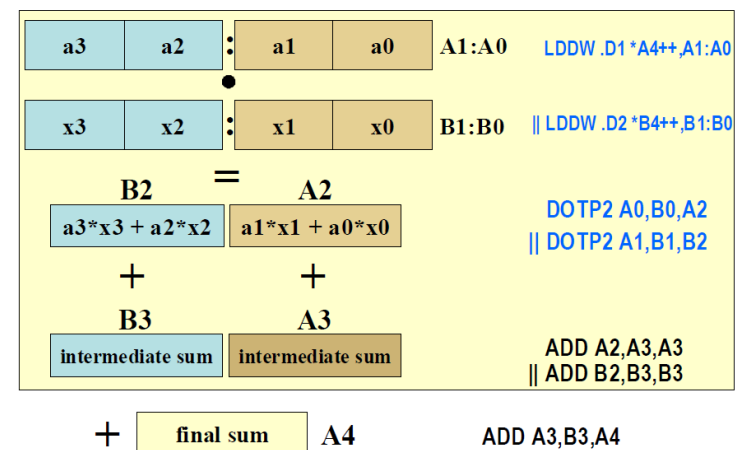
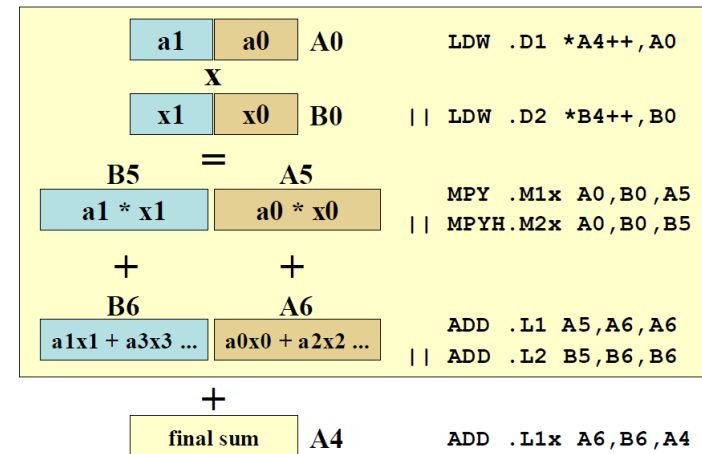
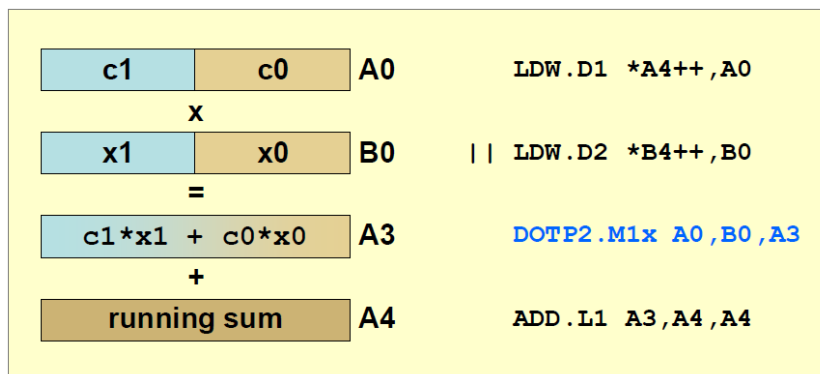
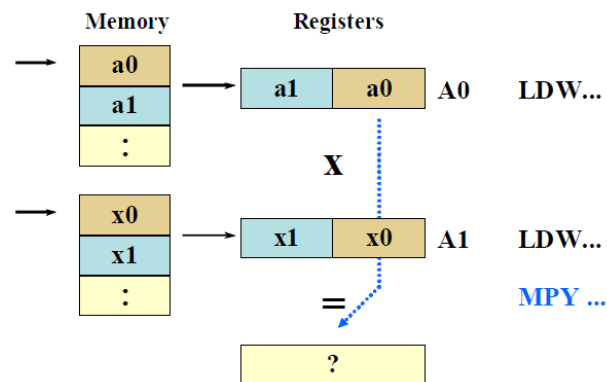
loop:
    ldh.d1  *A8++,A2
    || ldh.d1 *A9++,A3
    nop     4

    mpy.m1  A2,A3,A4
    nop
    add.l1  A4,A6,A6

    sub.l2  B0,1,B0
[b0] b.s1      loop
    nop     5
  
```

Software Design based on Embedded Processors

- ❖ An algorithm-level consideration of performance.
- ❖ Goal: Maximizing the utilization of function units.
- ❖ Using parallel instructions.



Software Design based on Embedded Processors

- ❖ An algorithm-level consideration of performance.
- ❖ Goal: Maximizing the utilization of function units.
- ❖ More advanced processors contain more powerful function units.

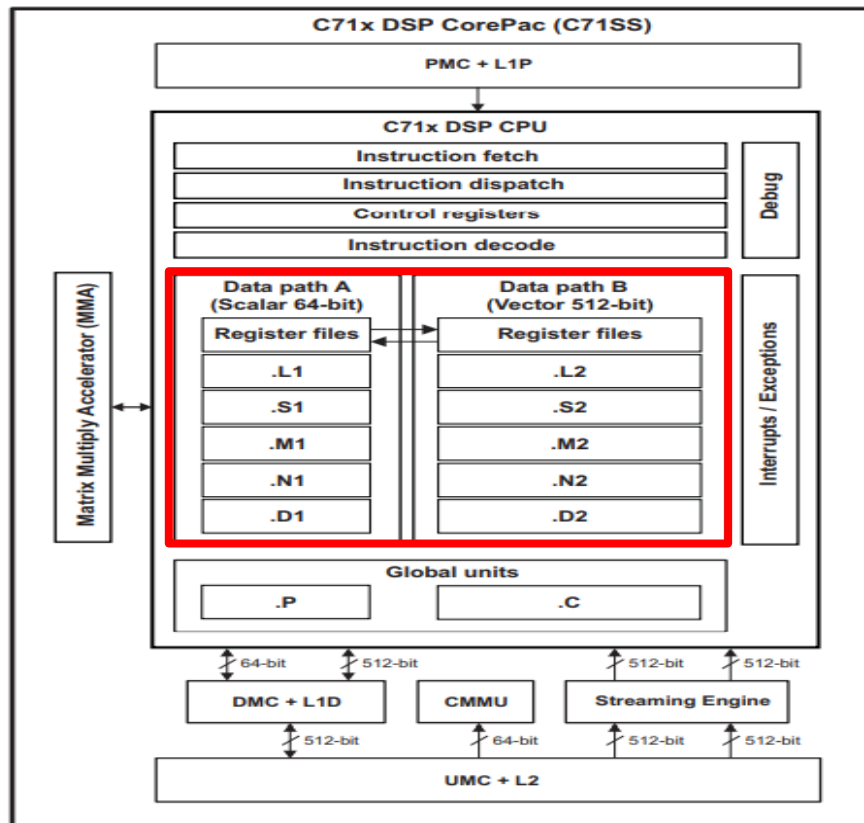


Figure 6-255. C71SS Overview

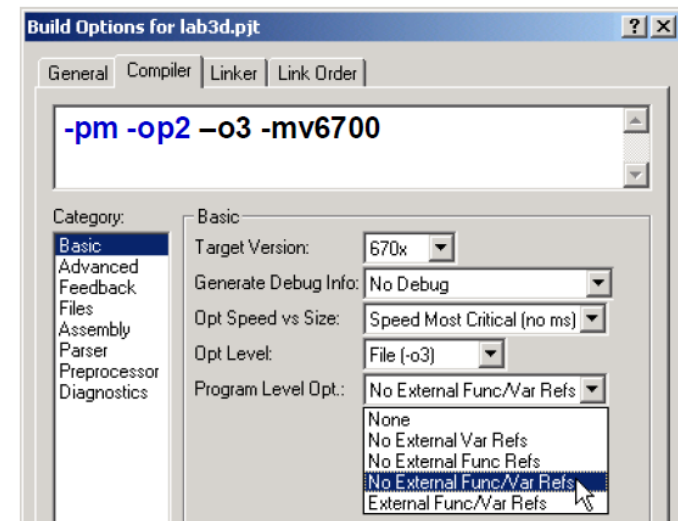
- **C7x-DSP** 浮點運算平台支持向量運算，在處理大量通用計算上與C6x DSP性能相比更強。
- 64位記憶體定址，單周期64位基本算術操作。
- 與C66相比，具有4-32倍的處理速度。
- 支援 MMA 特殊矩陣運算
- 支援全連接層（fc層）任意尺寸矩陣乘法運算
- 支援2D卷積（2d conv）
- 支援ReLU（relu）

Software Design based on Embedded Processors

- ❖ An algorithm-level consideration of performance.
- ❖ How do we invoke parallel instructions and maximize the utilization of function units?
 - Design your algorithm (Develop your C/C++ program) with architecture in mind.
 - Turn on compiler optimizers:

Program Level Optimization (-pm -o3)

- Provide compiler visibility to entire program



Software Design based on Embedded Processors

- ❖ An algorithm-level consideration of performance.
- ❖ How do we invoke parallel instructions and maximize the utilization of function units?
 - Using inline assembly/intrinsics:

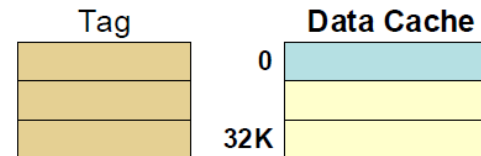
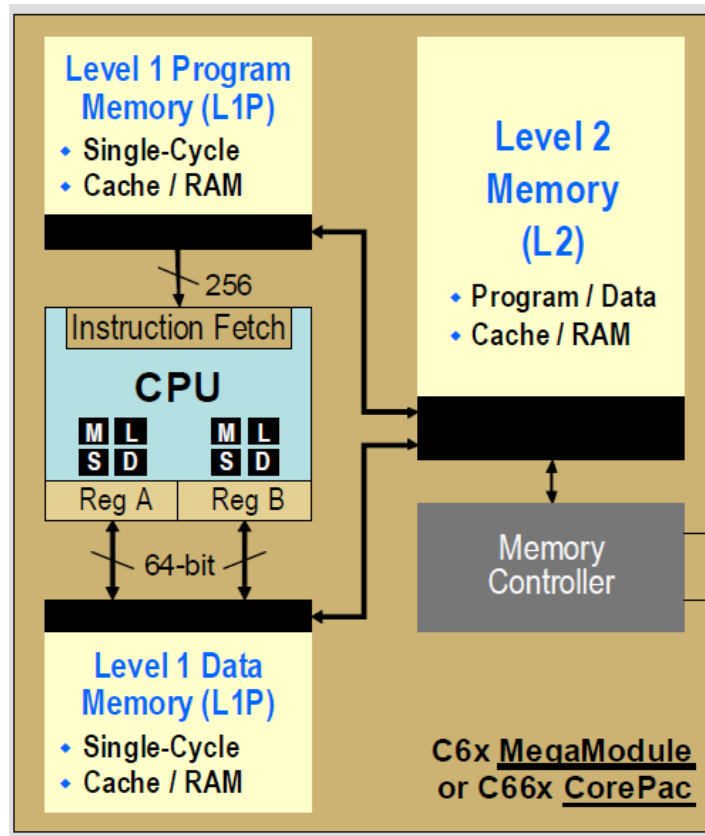
```
short a[50], b[50];
int y;

y = _add2(*(int *)a, *(int *)b);
```

for (i = 0; i < len; i += 4)	
{	
a3_a2 = _hill(_amemd8_const(&a[i]));	
a1_a0 = _loll(_amemd8_const(&a[i]));	B
b3_b2 = _hill(_amemd8_const(&b[i]));	
b1_b0 = _loll(_amemd8_const(&b[i]));	LDDW
/* Perform dot-products on pairs of elements, totaling the results in the accumulator. */	LDDW
sum_high += _dotp2(a3_a2, b3_b2);	DOTP2
sum_low += _dotp2(a1_a0, b1_b0);	ADD
}	DOTP2
	ADD

Software Design based on Embedded Processors

- ❖ A more system-level consideration of performance.
- ❖ Memory management with Cache behavior in mind.

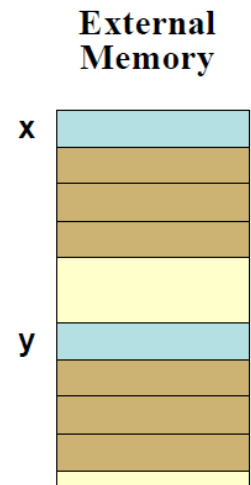


- ◆ One instruction may access multiple data elements:

```
for( i = 0; i < 4; i++ ) {
    sum += x[i] * y[i];
}
```

- ◆ What would happen if x and y ended up at the following addresses?

x = 0x8000
y = 0x9000



Software Design based on Embedded Processors

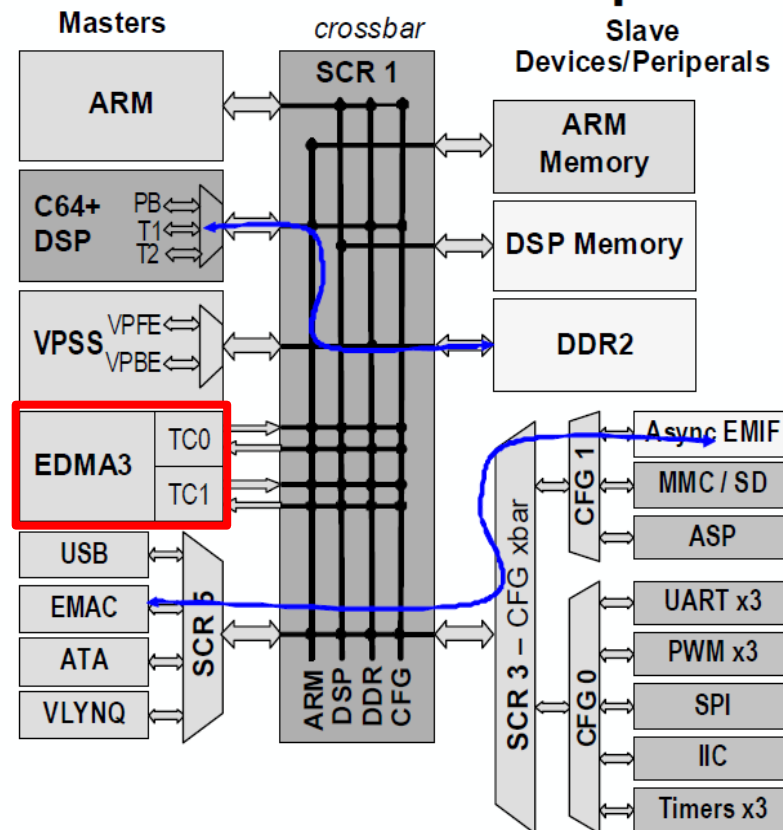
- ❖ A more system-level consideration of performance.
- ❖ Data movement with DMA (Direct Memory Access).

Master Devices:

- Can initiate data transfers

Slave Devices:

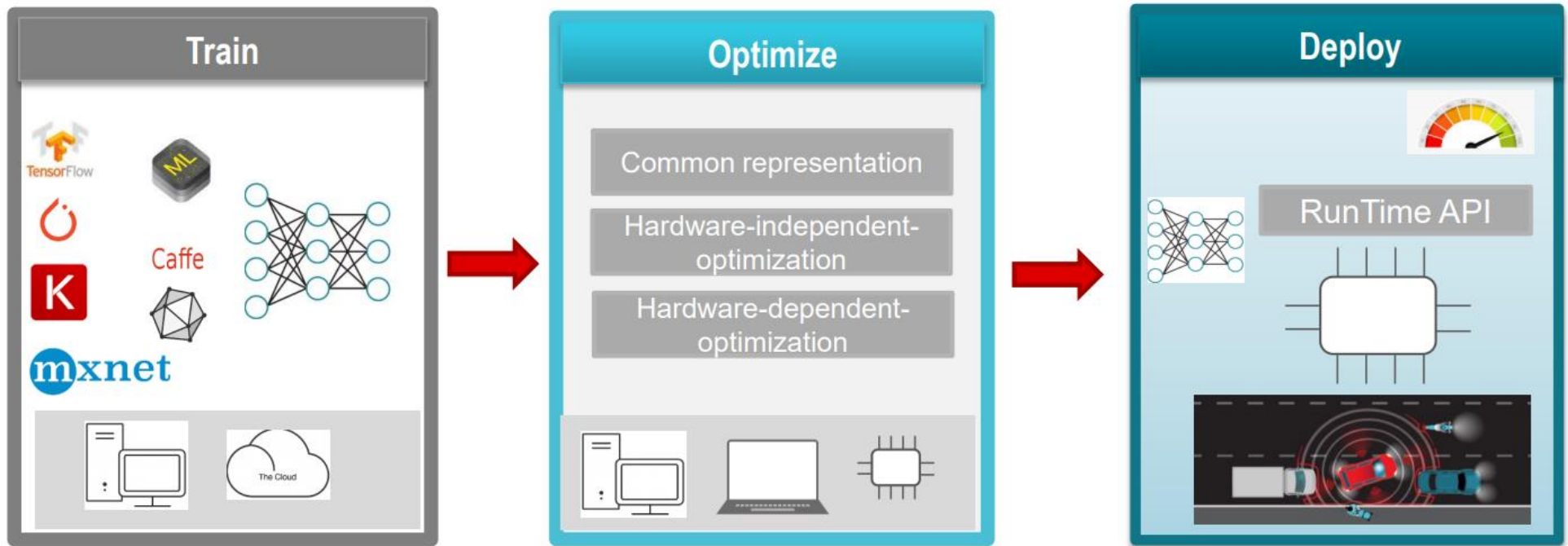
- Only Sink/Source for data transfers
- Can not initiate data transfers
- Often, they send an interrupt to request data transfer by CPU or EDMA3



➤ Moving data from one place to another without CPU intervene.

Edge Computing Deploying Data flow

■ TI Deep Learning (TIDL) high-performance fixed-point inference

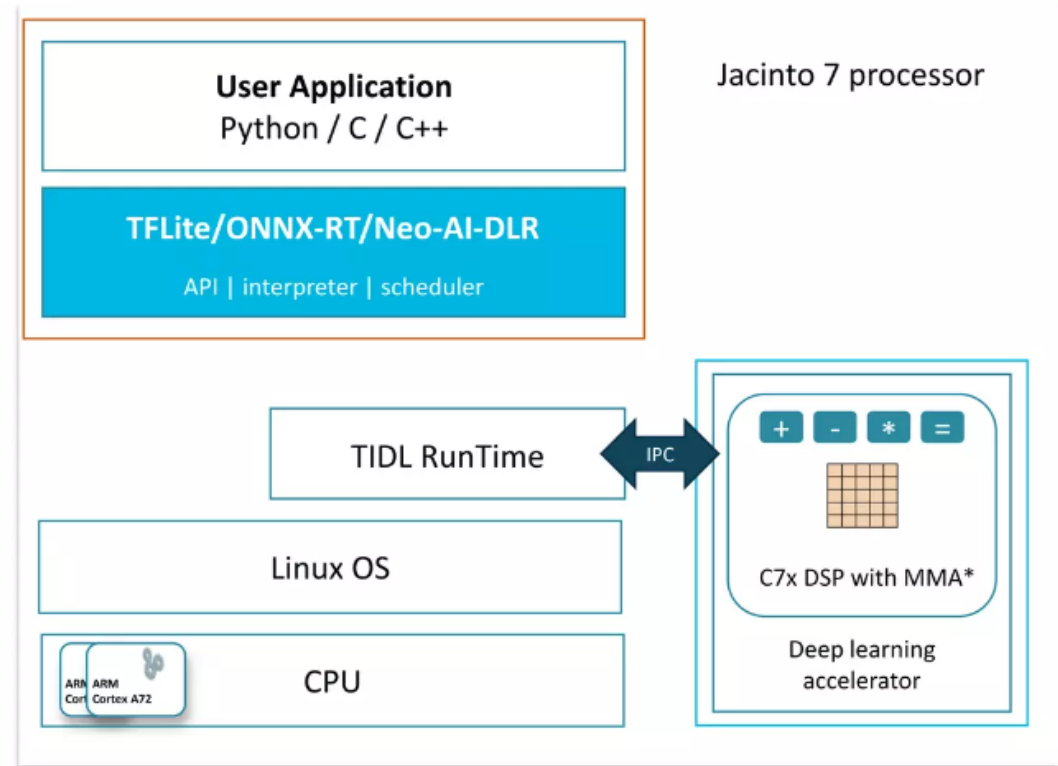
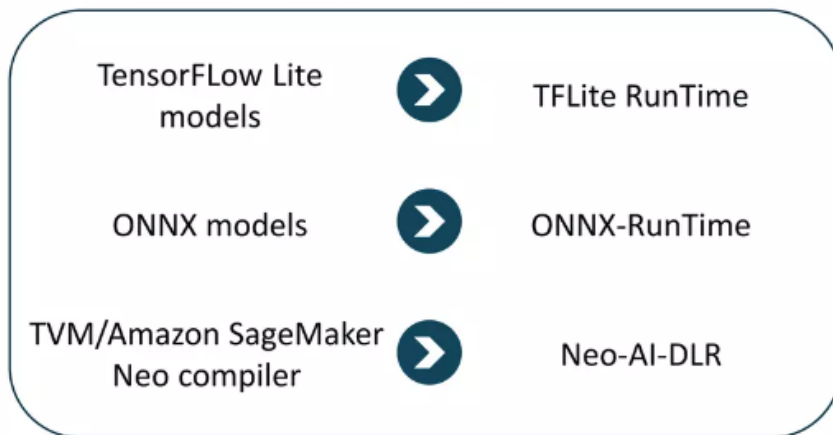


Edge Computing Deploying Data flow

■ Accelerate Models with Open source API

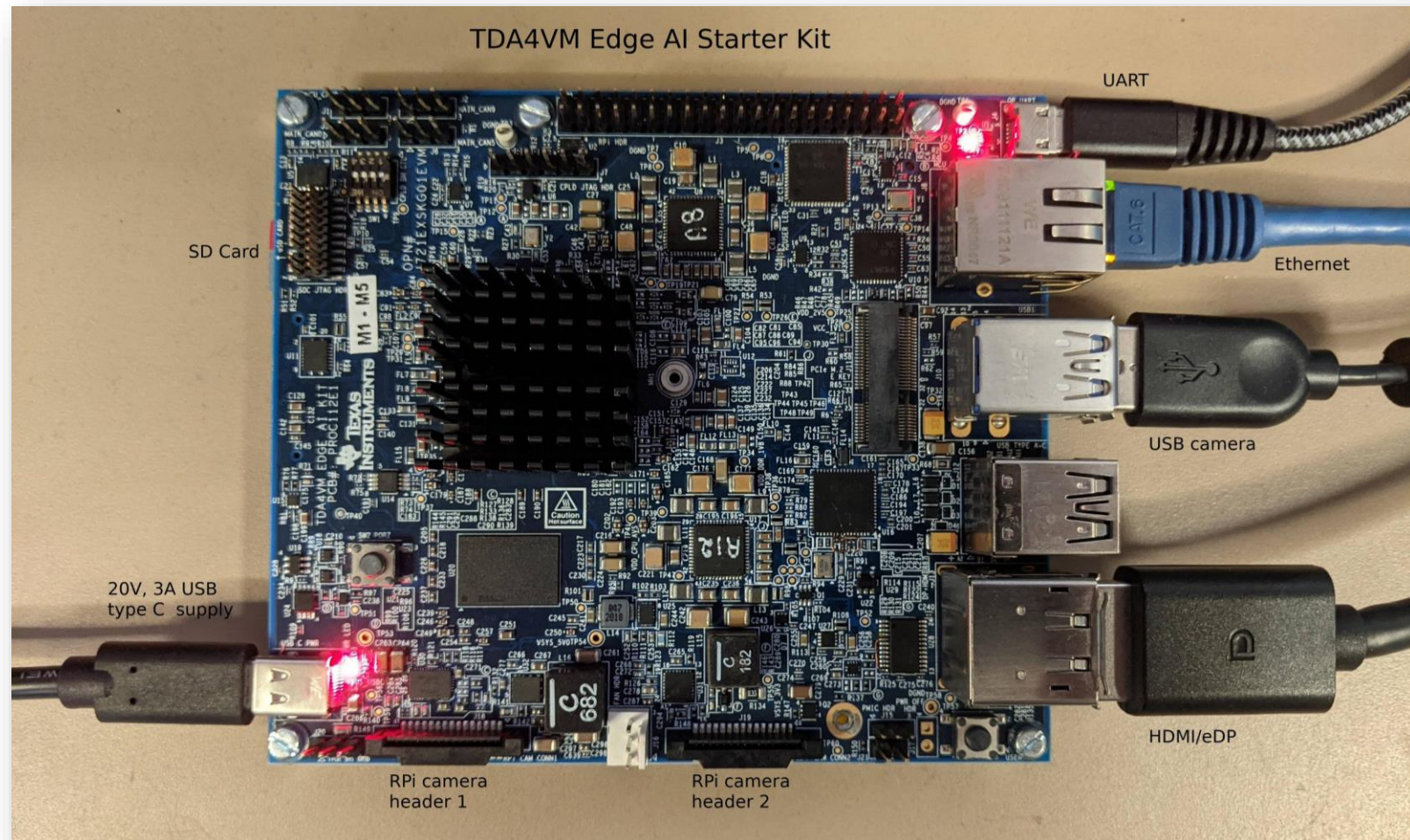
- Inference latency in <5 ms for all popular classification models
- Minimal accuracy loss with Post Training Quantization and Quantization Aware Training tools from TI

Run any model from open source inference frameworks on TI processors!



*MMA: Matrix Multiplication Accelerator (Tensor Processing Unit)

Overview



開發版環境安裝

下載檔案

❖ 網址

- <https://www.ti.com/tool/download/PROCESSOR-SDK-LINUX-SK-TDA4VM/09.00.00.08>
- SK-TDA4VM 簡介 – HackMD
 - <https://hackmd.io/nHhZREdtQiSqe35InAd2Yg?view>

❖ 下載TI網站內的「tisdk-edgeai-image-j721e-evm.wic.xz」開發板環境映像檔

↓ [tisdk-edgeai-image-j721e-evm.wic.xz](#) — 909583 K

SD card image with edge AI and robotics stack

MD5 checksum [fe93a552f9a702b6ee8a61d27a705cee](#)

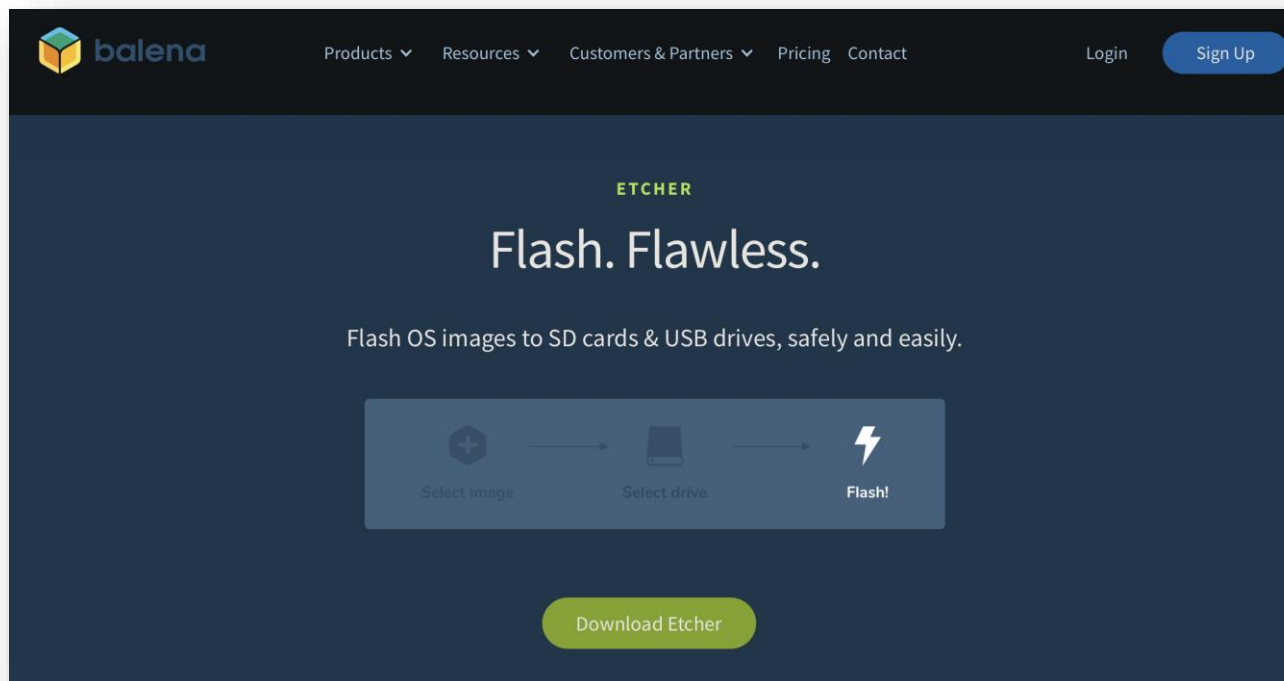


安裝燒錄軟體

❖ 網址

■ <https://etcher.balena.io>

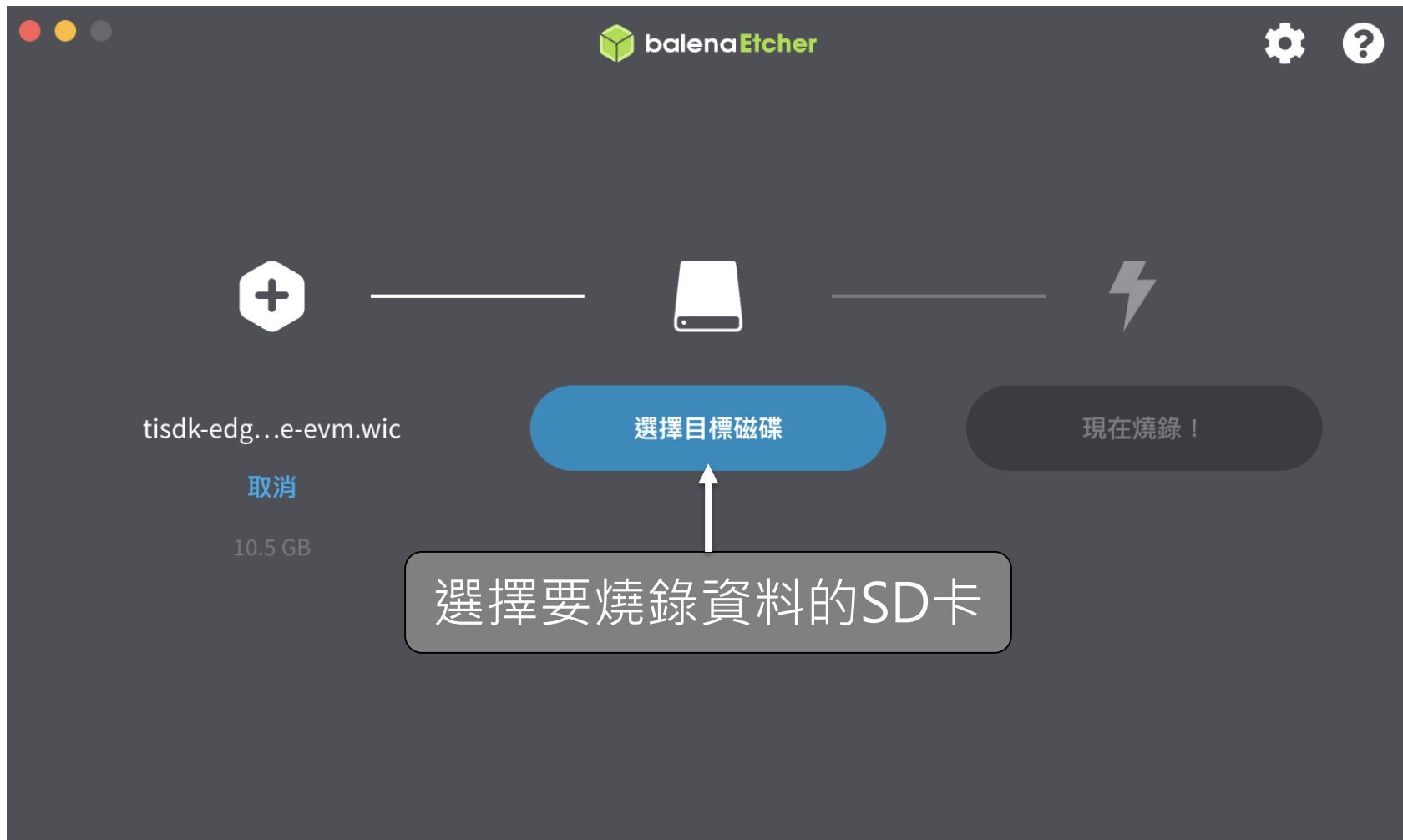
❖ 安裝balena etcher



燒錄SD



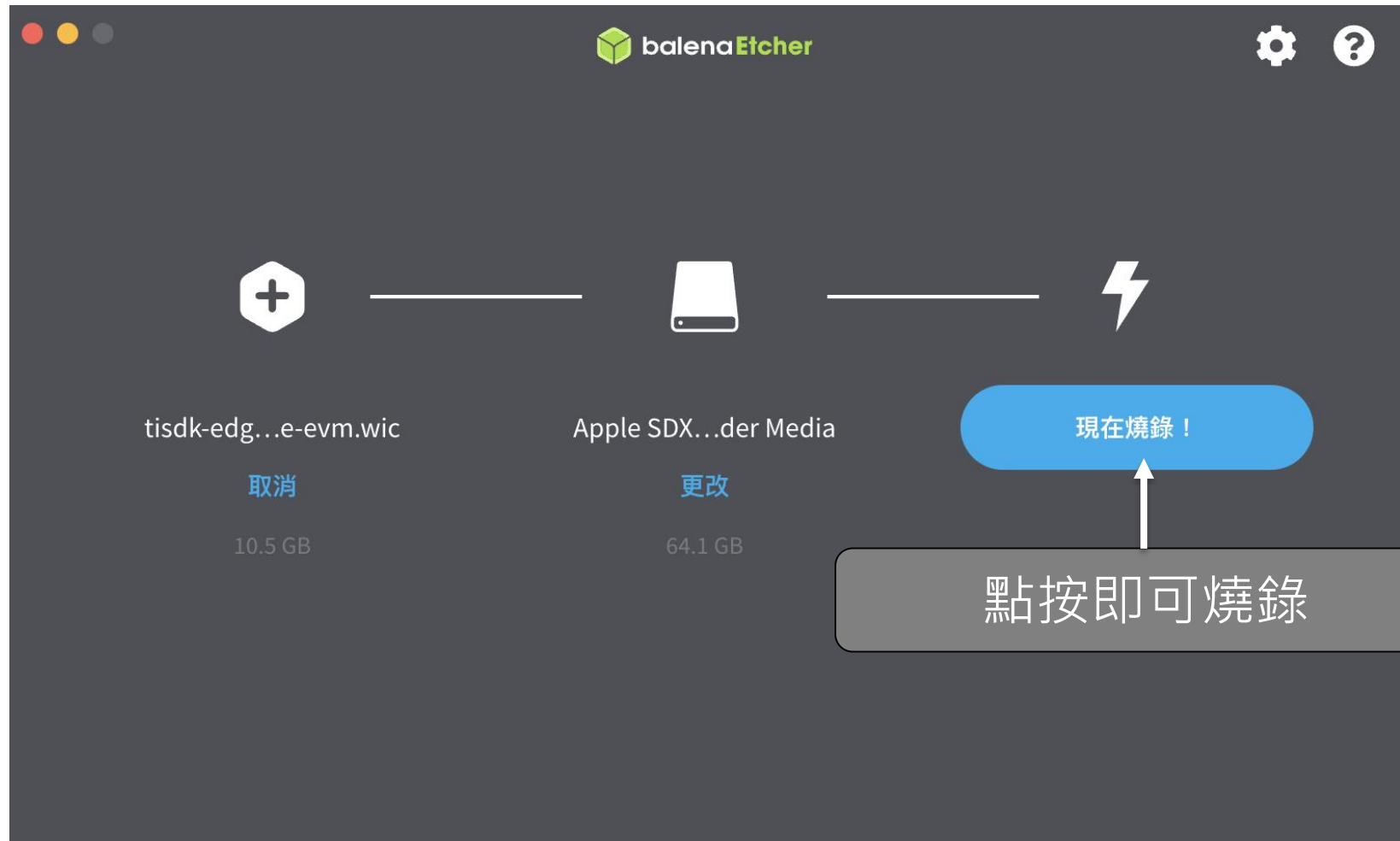
燒錄SD



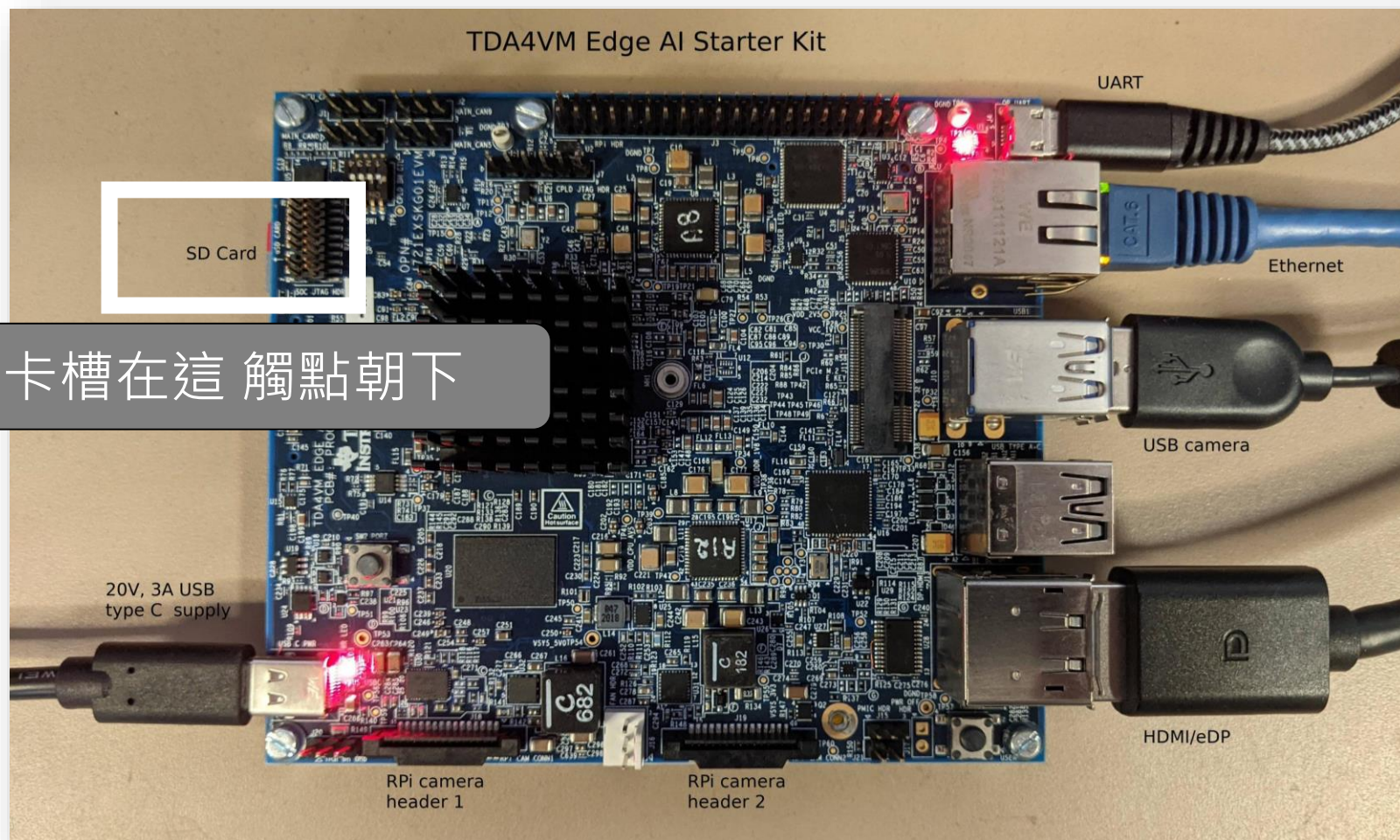
燒錄SD



燒錄SD

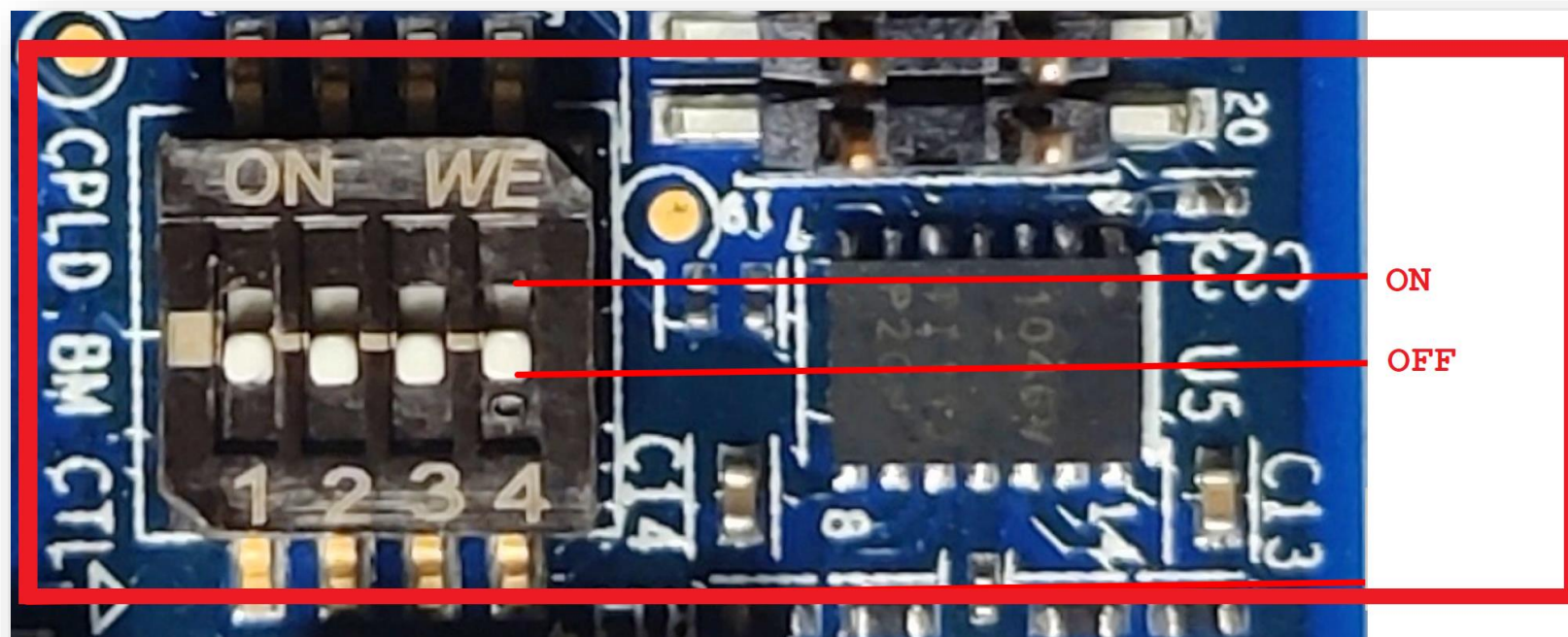


安裝SD卡

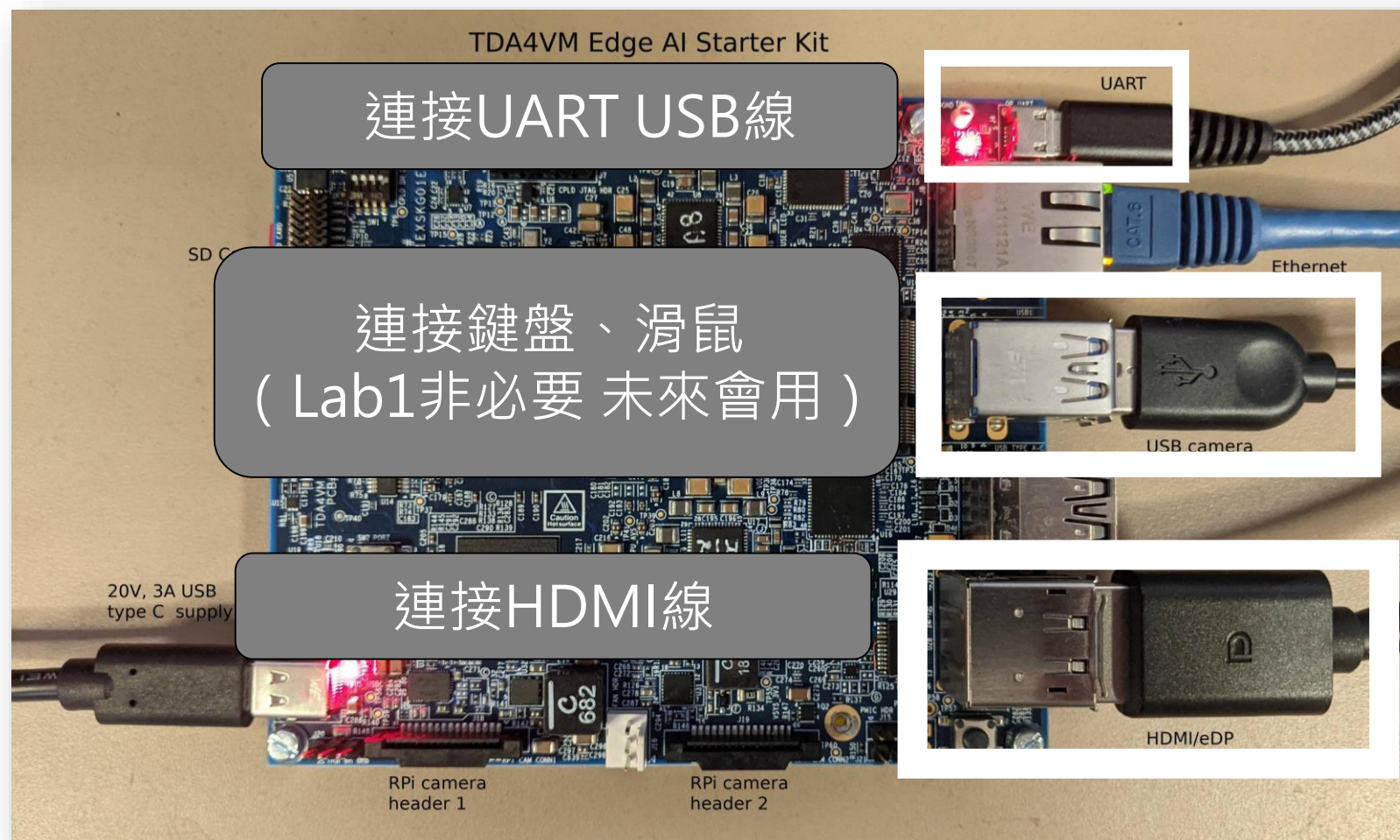


確認開機設定指撥開關

- ❖ 確認指撥開關如圖設定
- ❖ 開關就在SD卡位置附近



連接資料線



開發板操作

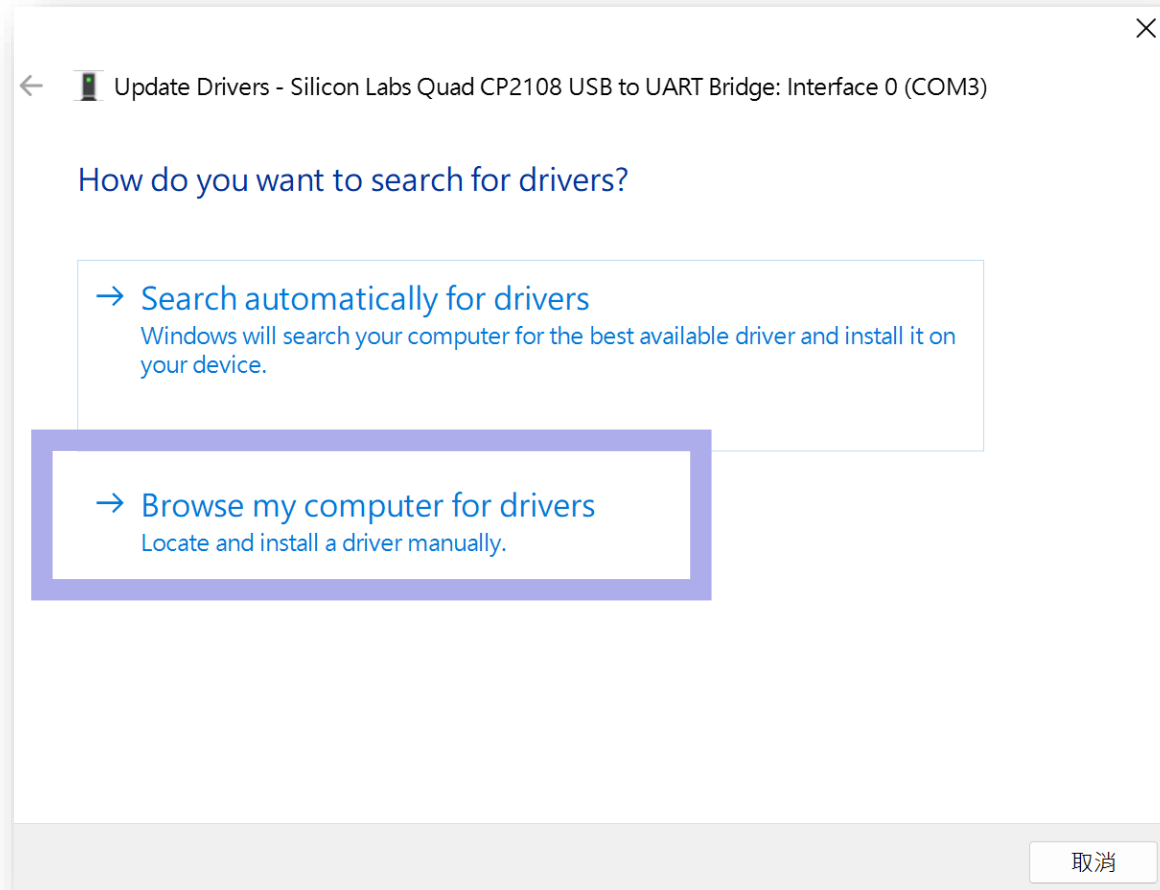
USB Serial驅動安裝

- ❖ 選擇Windows版本下載並解壓縮
 - <https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers>
- ❖ 開啟裝置管理員後，右鍵點選「其他裝置」內的 CP2108 Interface 0
- ❖ 點選「更新驅動程式」
- ❖ 安裝後拔除USB重新插上電腦，顯示下圖即安裝完成



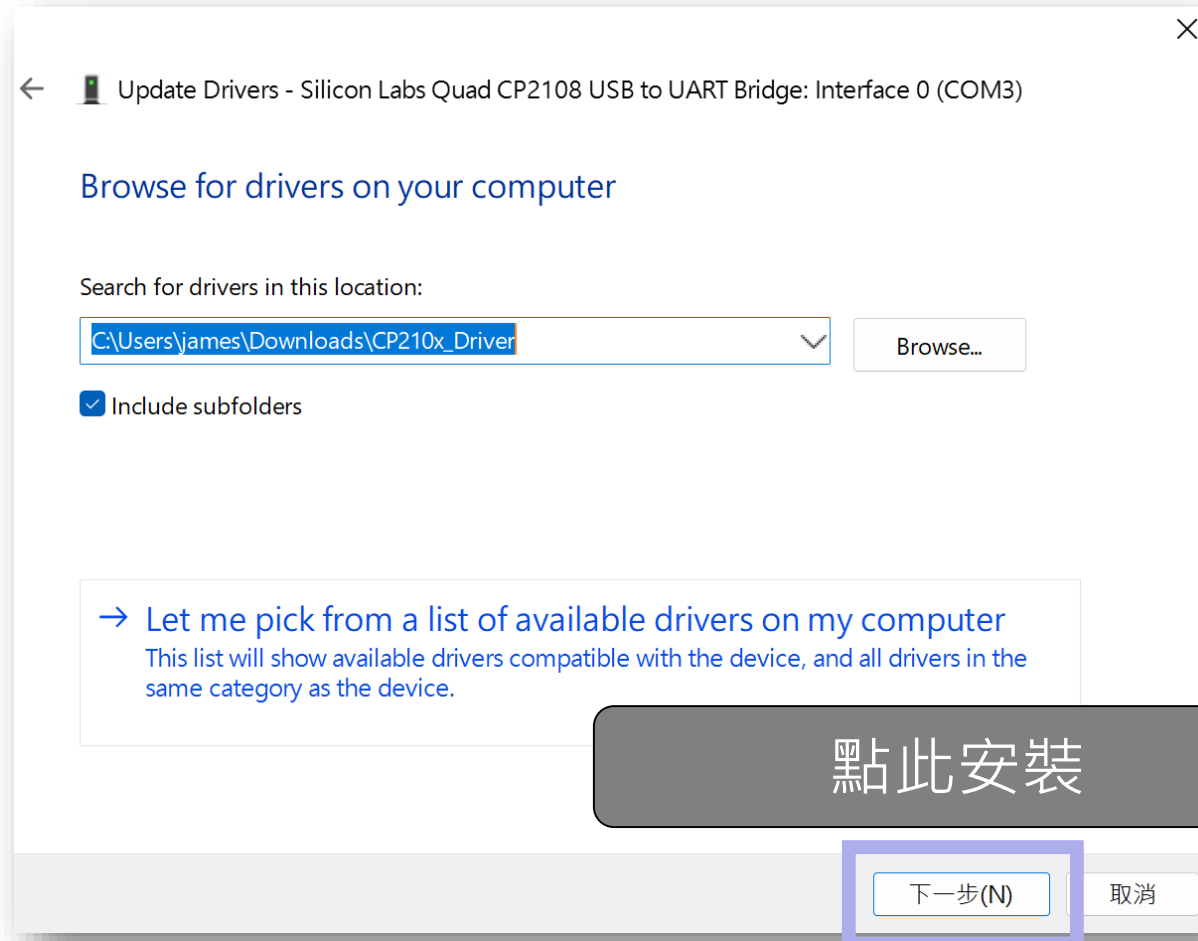
USB Serial驅動安裝

❖ 點選「瀏覽電腦上的驅動程式」



USB Serial驅動安裝

❖ 選擇下載的驅動程式資料夾，並點選包含子資料夾



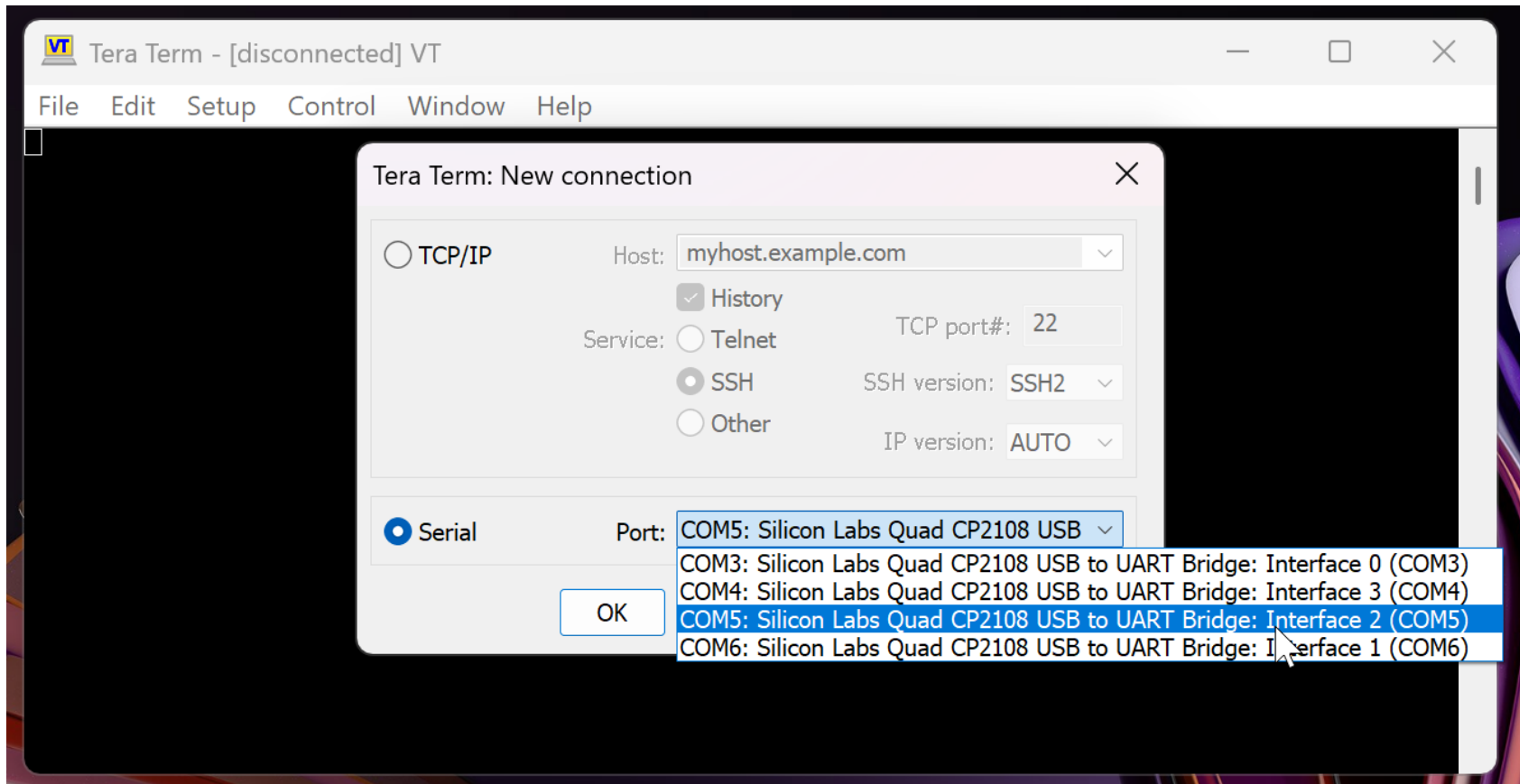
Tera Term

- ❖ 用來連接UART指令介面
- ❖ 其他序列埠程式（如PuTTY）也可以使用
[Releases · TeraTermProject/teraterm \(github.com\)](https://github.com/TeraTermProject/teraterm/releases)

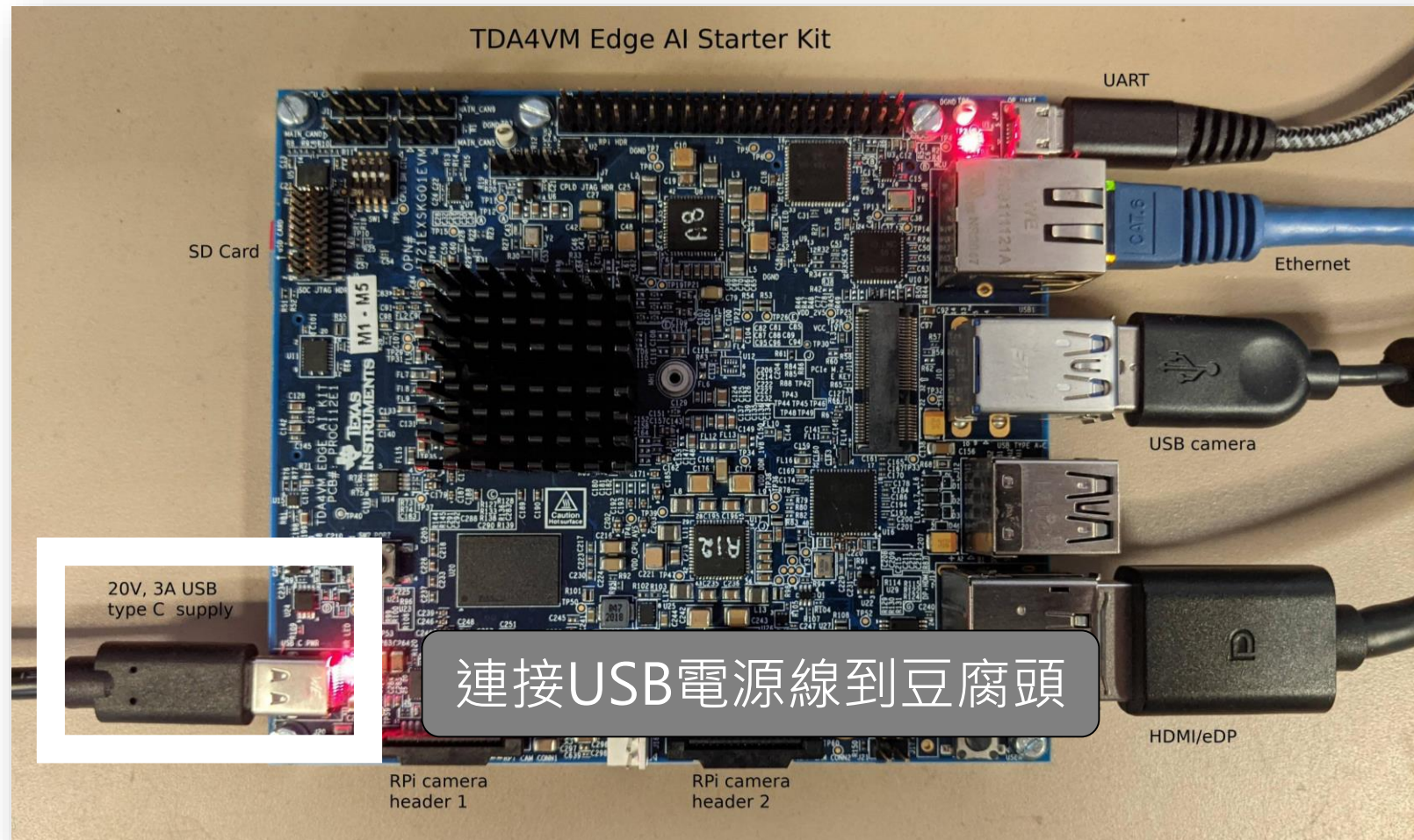


連接序列埠

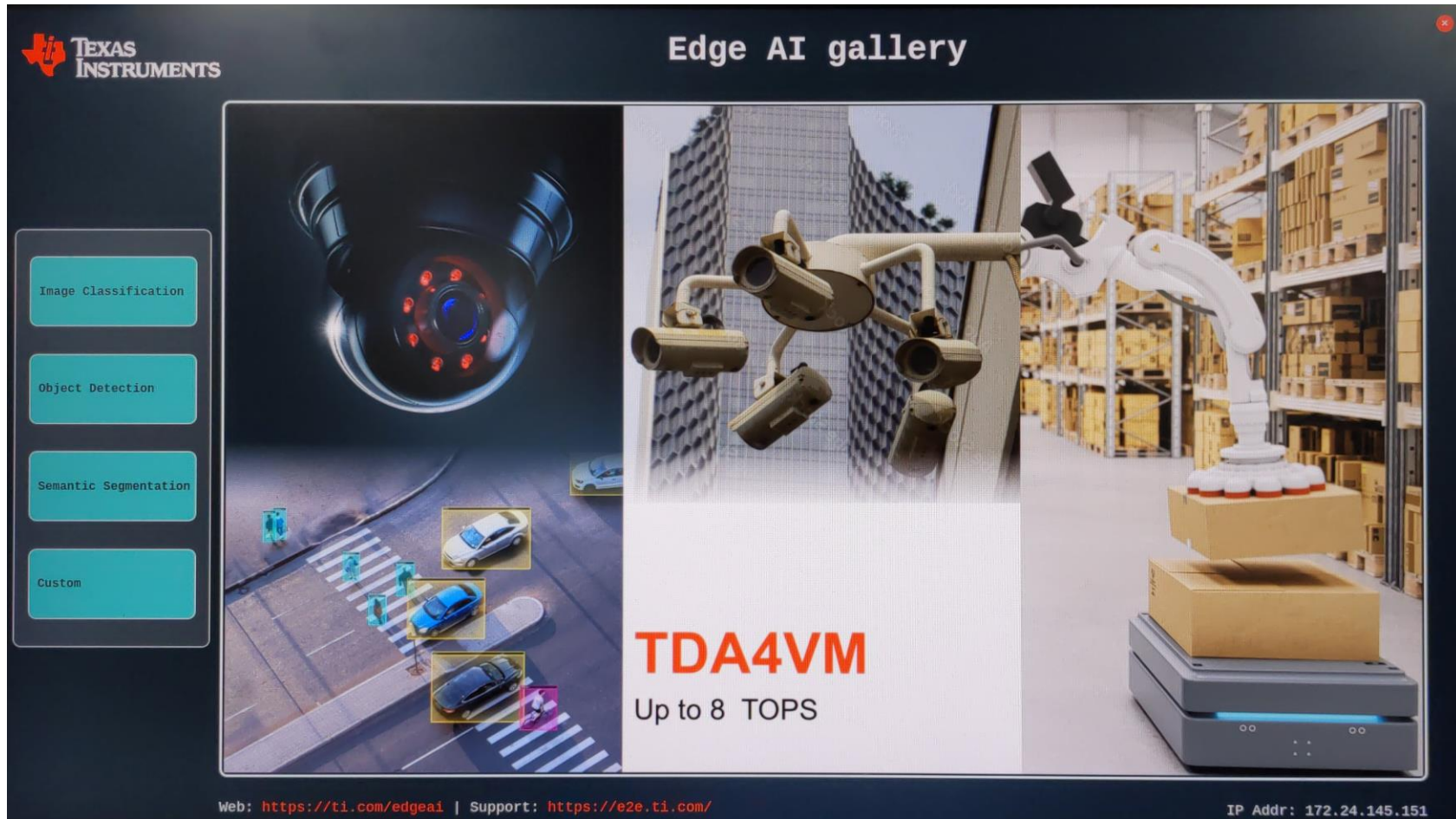
❖ 開啟Tera Term後選擇Interface 2(裝置管理員決定)



接電源

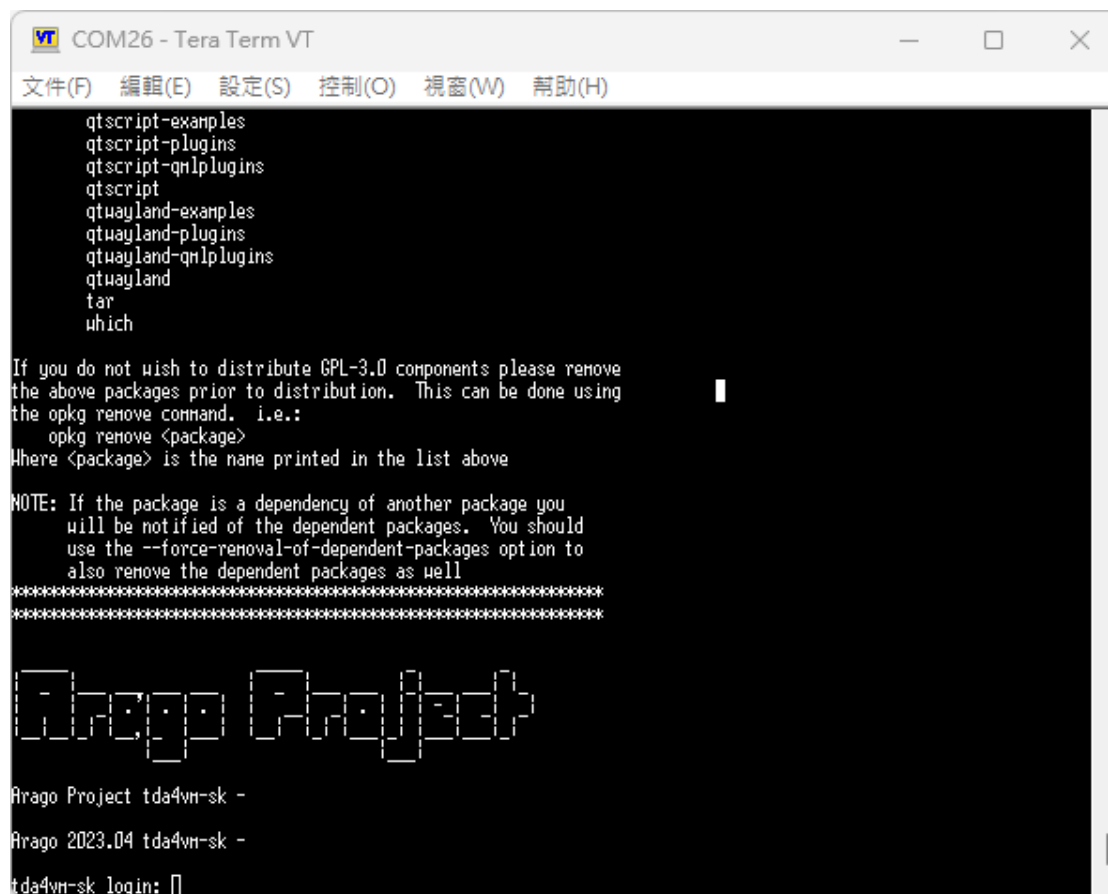


開機畫面



連接序列埠

❖ 成功連接畫面



```
COM26 - Tera Term VT
文件(F) 編輯(E) 設定(S) 控制(O) 視窗(W) 幫助(H)

qtscript-examples
qtscript-plugins
qtscript-qmlplugins
qtscript
qtwayland-examples
qtwayland-plugins
qtwayland-qmlplugins
qtwayland
tar
which

If you do not wish to distribute GPL-3.0 components please remove
the above packages prior to distribution. This can be done using
the opkg remove command. i.e.:
    opkg remove <package>
Where <package> is the name printed in the list above

NOTE: If the package is a dependency of another package you
will be notified of the dependent packages. You should
use the --force-removal-of-dependent-packages option to
also remove the dependent packages as well
*****
*****

Arago Project

Arago Project tda4vm-sk -
Arago 2023.04 tda4vm-sk -
tda4vm-sk login: 
```

登錄

❖ 輸入root登錄

❖ 成功畫面

```
COM26 - Tera Term VT
文件(F) 編輯(E) 設定(S) 控制(O) 視窗(W) 幫助(H)

will be notified of the dependent packages. You should
use the --force-removal-of-dependent-packages option to
also remove the dependent packages as well
*****
*****

Arago Project

Arago Project tda4vn-sk -
Arago 2023.04 tda4vn-sk -

tda4vn-sk login: root
[ 361.507522] audit: type=1006 audit(1651168890.628:10): pid=1259 uid=0 old-auid=4294967295 auid=0 tty=(none) old-
ses=4294967295 ses=3 res=1
[ 361.520406] audit: type=1300 audit(1651168890.628:10): arch=c00000b7 syscall=64 success=yes exit=1 a0=8 a1=ffffe
9328658 a2=1 a3=ffffbc7aa020 items=0 ppid=1 pid=1259 auid=0 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0
tty=(none) ses=3 comm="(systemd)" exe="/lib/systemd/systemd" key=(null)
[ 361.547129] audit: type=1327 audit(1651168890.628:10): proctitle="(systemd)"
[ 361.554460] audit: type=1334 audit(1651168890.640:11): prog-id=11 op=LOAD
[ 361.561586] audit: type=1300 audit(1651168890.640:11): arch=c00000b7 syscall=280 success=yes exit=8 a0=5 a1=ffff
ce37baf0 a2=78 a3=0 items=0 ppid=1 pid=1259 auid=0 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=(non
e) ses=3 comm="systemd" exe="/lib/systemd/systemd" key=(null)
[ 361.587246] audit: type=1327 audit(1651168890.640:11): proctitle="(systemd)"
[ 361.594581] audit: type=1334 audit(1651168890.640:12): prog-id=11 op=UNLOAD
[ 361.601652] audit: type=1334 audit(1651168890.640:13): prog-id=12 op=LOAD
[ 361.608662] audit: type=1300 audit(1651168890.640:13): arch=c00000b7 syscall=280 success=yes exit=8 a0=5 a1=ffff
ce37bb90 a2=78 a3=0 items=0 ppid=1 pid=1259 auid=0 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=(non
e) ses=3 comm="systemd" exe="/lib/systemd/systemd" key=(null)
[ 361.634217] audit: type=1327 audit(1651168890.640:13): proctitle="(systemd)"
root@tda4vn-sk:/opt/edgeai-gst-apps#
```

如何使用VI文字編輯器

- ❖ 在終端機（文字指令）介面中無法使用GUI編輯器，但可以使用VI等文字介面編輯器
- ❖ 先備知識：Linux（UNIX系列）資料路徑表示
 - 以斜線開頭代表絕對路徑（Windows用反斜線\）
 - 例：`/home/user/Downloads/main.c`
 - 以`.`或`..`開頭代表相對路徑
 - 「`..`」代表目前所在資料夾的前一個資料夾
 - 假設你在`/home/user`，`../`就代表`/home`
 - 「`.`」代表目前所在資料夾
 - 例：`../Downloads/main.c`
 - 例：`./main.c`

如何使用VI文字編輯器

❖ 先備知識：Linux操作/查看目前所在資料夾

- 指令從:到#字號中間是目前所在資料夾路徑

```
root@tda4vm-sk:/opt/edgeai-gst-apps#
```

- 相對路徑就是相對於目前所在資料夾
- cd指令
 - 改變目前所在資料夾到目標資料夾
 - 格式: **cd** <目標資料夾路徑>
- ls指令
 - 查看目前所在資料夾或指定資料夾內容(檔案)
 - 目前資料夾(後面空白): **ls**
 - 指定資料夾: **ls** <指定資料夾路徑>

如何使用VI文字編輯器

- ❖ 在終端機打上**vi** <文件路徑>即可開啟該文件
- ❖ 例：**vi ./test.txt**
- ❖ 若文件不存在會新增(但資料夾不存在會錯誤)
- ❖ 剛開啟時無法輸入文字，此時是命令模式
- ❖ 按下**i**或**a**編輯檔案
- ❖ 按下**Esc**回到命令模式



如何使用VI文字編輯器

- ❖ 編輯好文字後，按下Esc回到命令模式
- ❖ 輸入冒號來打指令，游標會跑到最下面
- ❖ q為不儲存離開（未更改文件時才能用）
- ❖ wq為儲存並離開
- ❖ q!為不儲存強制離開



The screenshot shows a terminal window titled "COM26 - Tera Term VT". The menu bar includes "文件(F)", "編輯(E)", "設定(S)", and "控制(C)". The main text area displays "Hello TDA4VM!" followed by several lines of tilde (~) characters. At the bottom, the prompt ":" is visible, indicating the VI editor is in command mode, with the command "wq" being entered.

Lab1: 跑一個小型的矩陣運算DSP程式

Winograd Convolution Matrix Multiply Accelerate (MMA)

```
COM11 - PuTTY
root@tda4vm-sk:~# vi mma.cpp
root@tda4vm-sk:~# g++ mma.cpp -o mma
root@tda4vm-sk:~# ./mma
Result of Winograd matrix multiplication:
84 90 96
201 216 231
318 342 366
root@tda4vm-sk:~#
```

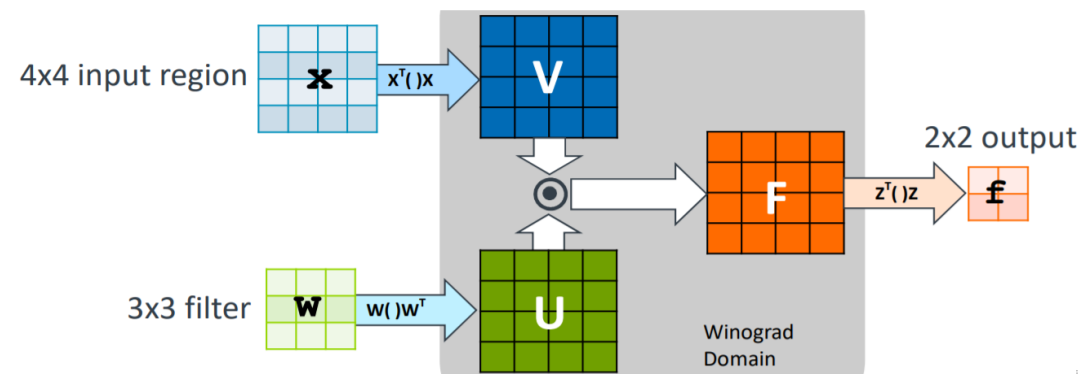
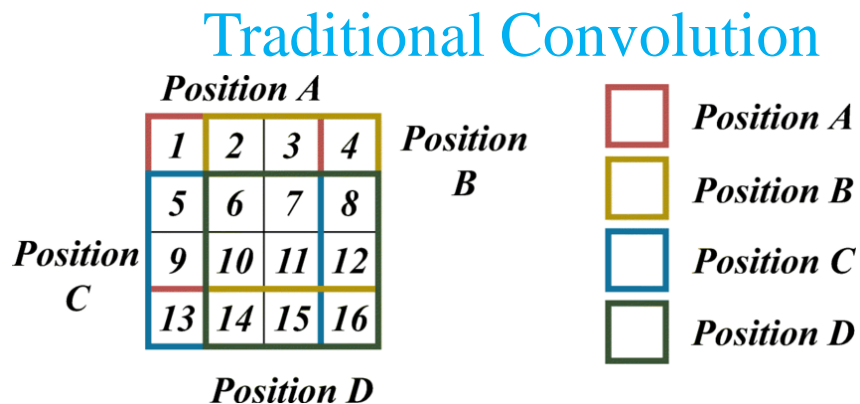
如有error: `gcc mma.cpp -lstdc++ -o mma`

$$F(2,3) = \begin{bmatrix} d0 & d1 & d2 \\ d1 & d2 & d3 \end{bmatrix} \begin{bmatrix} g0 \\ g1 \\ g2 \end{bmatrix} = \begin{bmatrix} m1 + m2 + m3 \\ m2 - m3 - m4 \end{bmatrix}$$

$$m1 = (d0 - d2)g0 \quad m2 = (d1 + d2) \frac{g0 + g1 + g2}{2}$$

$$m4 = (d1 - d3)g2 \quad m3 = (d2 - d1) \frac{g0 - g1 + g2}{2}$$

Winograd Convolution



Lab1: 跑一個小型的矩陣運算DSP程式

■ Winograd Convolution C Code

```

1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  void WinogradMultiply(vector<vector<int>>& A, vector<vector<int>>& B, vector<vector<int>>& C, int n) {
7      vector<int> row_factor(n), col_factor(n);
8
9      // Precompute Row Factor for A
10     for(int i = 0; i < n; i++) {
11         row_factor[i] = A[i][0] * A[i][1];
12         for(int j = 1; j < n/2; j++) {
13             row_factor[i] = row_factor[i] + A[i][2 * j] * A[i][2 * j + 1];
14         }
15     }
16
17     // Precompute Column Factor for B
18     for(int i = 0; i < n; i++) {
19         col_factor[i] = B[0][i] * B[1][i];
20         for(int j = 1; j < n/2; j++) {
21             col_factor[i] = col_factor[i] + B[2 * j][i] * B[2 * j + 1][i];
22         }
23     }
24
25     // Initialize matrix C to zero
26     for(int i = 0; i < n; i++) {
27         for(int j = 0; j < n; j++) {
28             C[i][j] = 0;
29         }
30     }
31
32     // Compute the product
33     for(int i = 0; i < n; i++) {
34         for(int j = 0; j < n; j++) {
35             C[i][j] = -row_factor[i] - col_factor[j];
36             for(int k = 0; k < n / 2; k++) {
37                 C[i][j] += (A[i][2 * k] + B[2 * k + 1][j]) * (A[i][2 * k + 1] + B[2 * k][j]);
38             }
39         }
40     }
41
42     // If n is odd, add the contributions of the odd rows and columns
43     if(n % 2 != 0) {
44         for(int i = 0; i < n; i++) {
45             for(int j = 0; j < n; j++) {
46                 C[i][j] += A[i][n - 1] * B[n - 1][j];
47             }
48         }
49     }
50 }
51
52 int main() {
53     // Define 3x3 matrices A, B and C
54     vector<vector<int>> A = {{1, 2, 3},
55                             {4, 5, 6},
56                             {7, 8, 9}};
57
58     vector<vector<int>> B = {{10, 11, 12},
59                             {13, 14, 15},
60                             {16, 17, 18}};
61
62     vector<vector<int>> C(3, vector<int>(3, 0));
63
64     // Perform matrix multiplication
65     WinogradMultiply(A, B, C, 3);
66
67     // Print the result
68     cout << "Result of Winograd matrix multiplication:" << endl;
69     for(int i = 0; i < 3; i++) {
70         for(int j = 0; j < 3; j++) {
71             cout << C[i][j] << " ";
72         }
73         cout << endl;
74     }
75     return 0;
76 }

```

參考資料與文獻

- [1] [TDA4VM Processors datasheet \(Rev. K\)](#)
- [2] [J721E DRA829/TDA4VM Processors Silicon Revision 1.1/1.0 \(Rev. D\)](#)
- [3] [DRA829/TDA4VM Technical Reference Manual \(Rev. C\)](#)
- [4] [Jacinto7 AM6x, TDA4x, and DRA8x High-Speed Interface Design Guidelines \(Rev. A\)](#)
- [5] [TMS320C6652 and TMS320C6654 Fixed and Floating-Point Digital Signal Processor datasheet \(Rev. E\)](#)
- [6] [TMS320C6652/54/55/57 Multicore Fixed and Floating-Point DSP SR1.0 \(Rev. C\)](#)
- [7] [SK-TDA4VM User's Guide \(Rev. D\)](#)
- [8] [J721EXSKG01EVM EU Declaration of Conformity \(DoC\) \(Rev. A\)](#)
- [9] [DMA Controller Module \(Chapter Excerpt From MSP430x5xx Family, SLAU208\) \(Rev. F\)](#)
- [10] [AM437x Sitara™ Processors datasheet \(Rev. E\)](#)
- [11] [AM437x and AMIC120 ARM® Cortex™-A9 Processors Technical Reference Manual \(Rev. I\)](#)
- [12] [TMS320C6000 DSP Cache User's Guide \(Rev. A\)](#)