數位系統導論實驗

Lab9

Squarer on FPGA

負責助教: 蔣宗廷

Outline

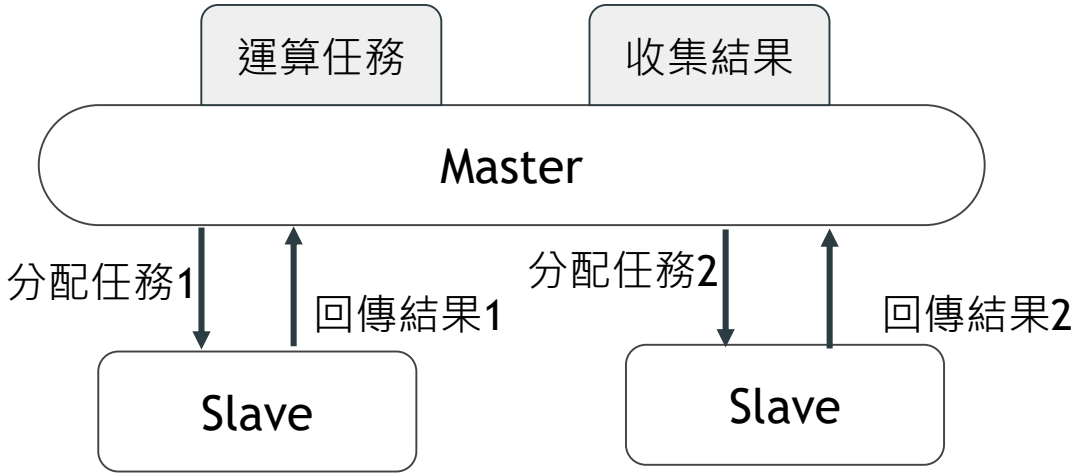
- 課程目的
- Master-slave co-processing 介紹
- Step-by-Step 範例操作
- 範例說明
- 作業內容與評分方式
- 附錄 (Vivado安裝、創建新檔)

課程目的

- 透過範例讓同學
  - 正確安裝 Vivado開發套件
  - 了解燒錄 Zedboard的流程
  - 以 FPGA 實現運算並透過C code 控制

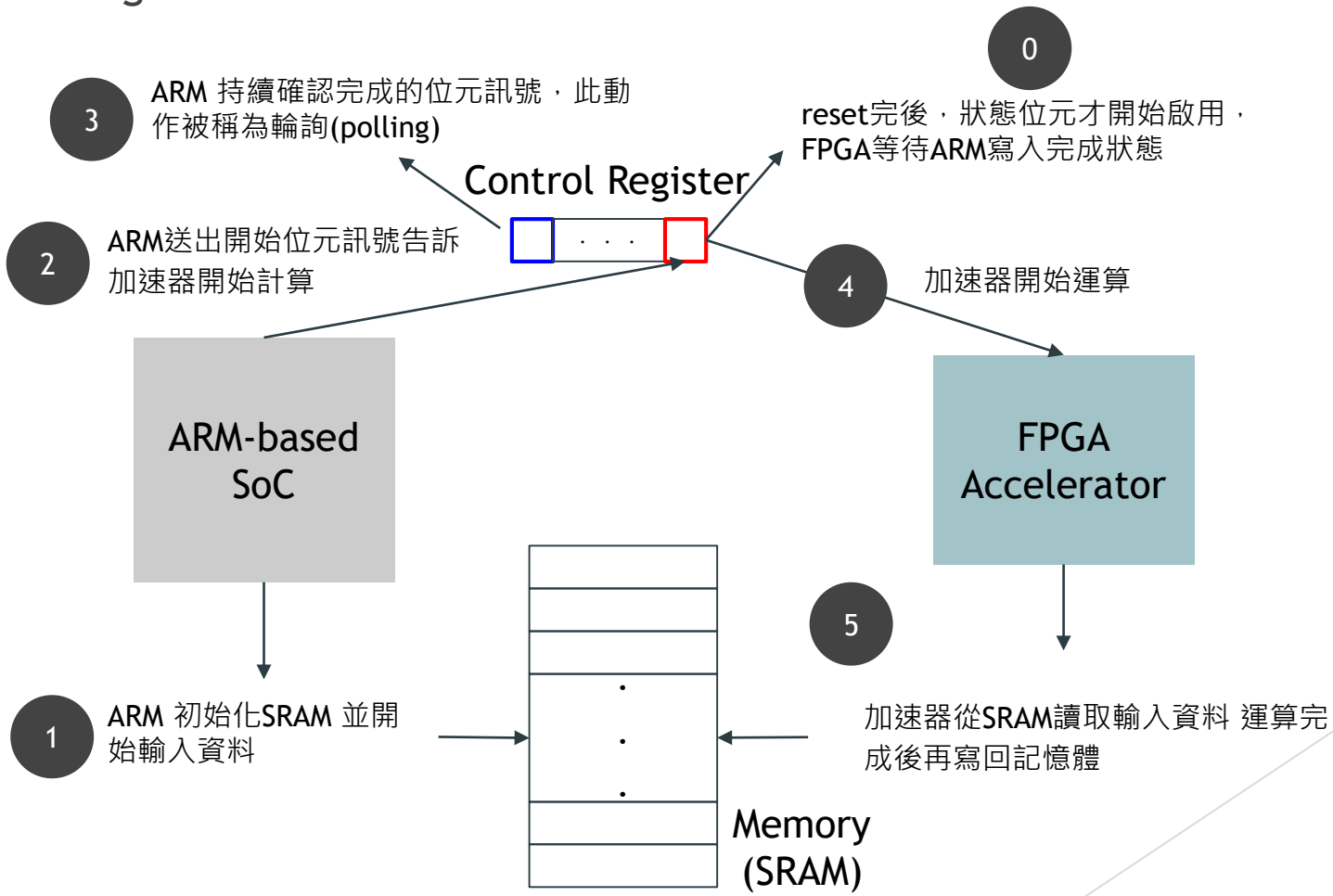
Master-slave co-processing 介紹(1/3)

- Master-slave
  - 在Master-slave中有一台機器作為Master，其他機器作為Slave，而Master作為任務分配者，給多個Slave分配計算任務，最後由Master匯集結果。



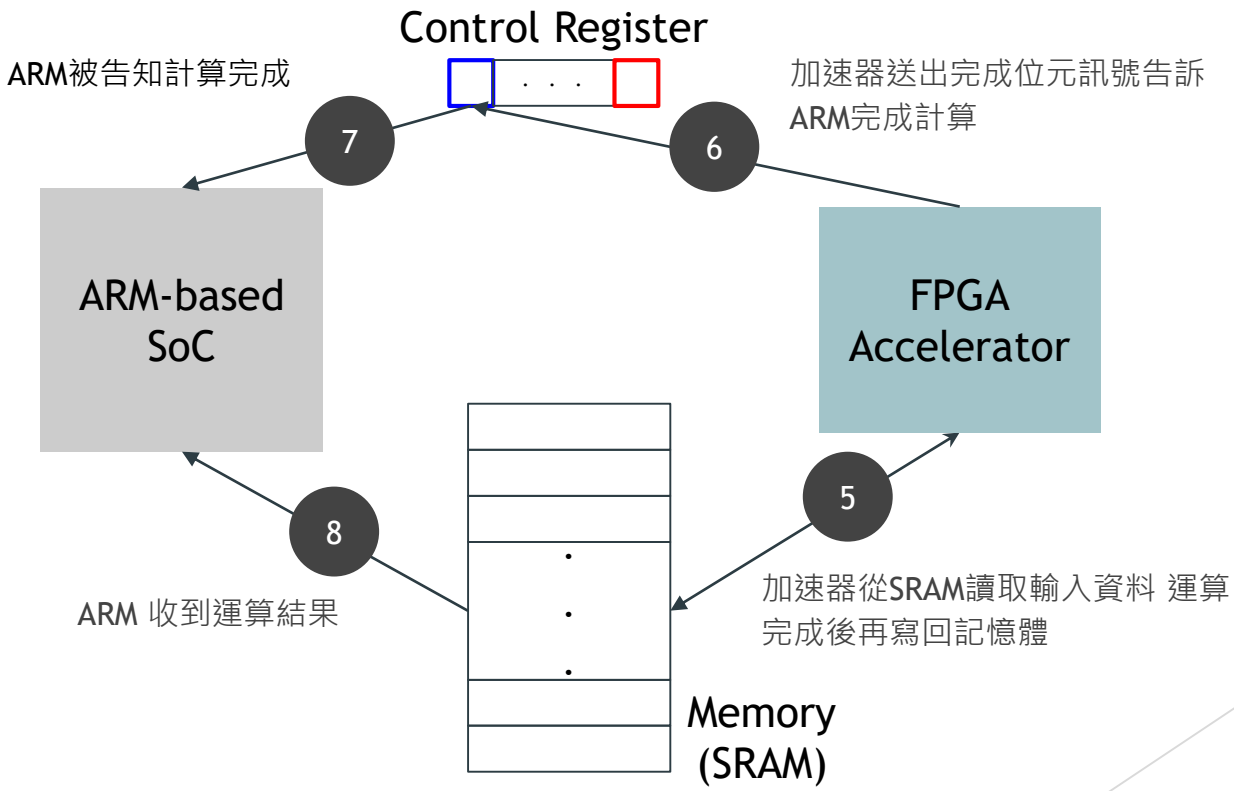
Master-slave co-processing 介紹(2/3)

- Co-processing流程



Master-slave co-processing 介紹(3/3)

- Co-processing流程

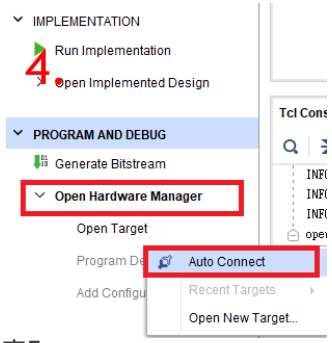


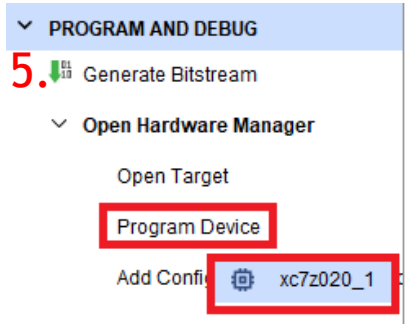
範例操作 -編譯 C for ARM

- 解壓縮arm-2012.03-57-arm-none-linux-gnueabi-i686-mingw32
- 新增PATH環境變數:XXX\arm\arm-2012.03\bin
- 打開cmd 輸入指令: arm-none-linux-gnueabi-gcc lab9.c -o lab9
- 將lab9 執行檔放入SD卡

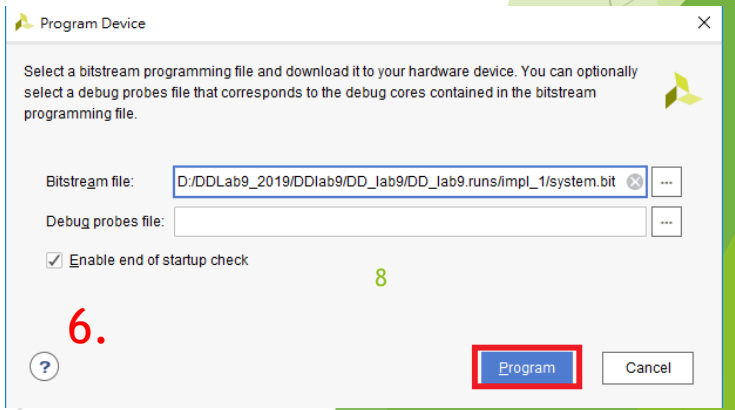
範例操作 - 燒錄至FPGA

- Step1：參照附錄, 安裝Vivado
- Step2：將USB線連接Zedboard PROG port與電腦
- Step3：開啟DD_lab9.xpr
- Step4：點選左下角Open Hardware Manager再點擊Open target並選擇Auto Connect





- Step5：點擊Program Device，並選xc7z020_1
- Step6：燒錄程式到FPGA上



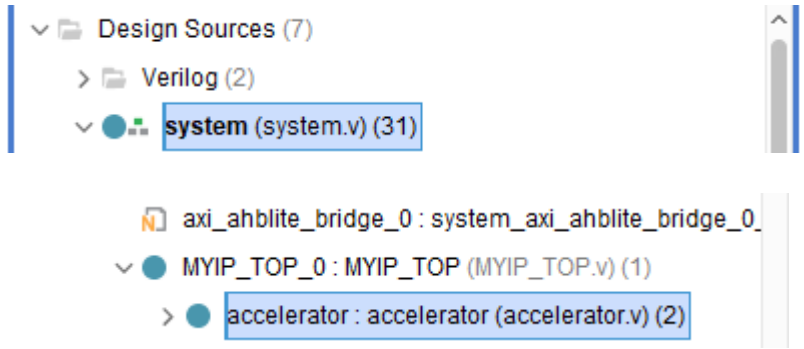
範例操作 – 於Terminal輸出結果

- 開啟MobaXTerm，並掛載file system
  - mount /dev/mmcblk0p1 /mnt
- 將目錄切換到SD卡
  - cd /mnt
- 設定lab9的權限為可執行
  - chmod +x lab9
- 執行lab9
  - ./lab9

```
zynq> chmod +x lab9
zynq> ./lab9
Zynq_BASE mapping successful :
0x80000000 to 0xb6f81000, size = 1024
Original values:
Input  0 =                          3f000000
Input  1 =                          3e99999a
Input  2 =                          3e800000
Input  3 =                          3f4ccccd
Input  4 =                          3f19999a
Input  5 =                          3fc00000
Input  6 =                          3fa00000
Input  7 =                          3f400000
Input  8 =                          40600000
Input  9 =                          40900000
Input 10 =                          41200000
Input 11 =                          40280000
Input 12 =                          3fe00000
Input 13 =                          40f80000
Input 14 =                          41b20000
Input 15 =                          410c0000
// ----------------------------Running!-----------------------------------//
start run
polling busy
operation finish
// ----------------------------Finish!------------------------------------//
Values after caculation:
answer  0 =                          3e800000
answer  1 =                          3db851ec
answer  2 =                          3d800000
answer  3 =                          3f23d70a
answer  4 =                          3eb851ec
answer  5 =                          40100000
answer  6 =                          3fc80000
answer  7 =                          3f100000
answer  8 =                          41440000
answer  9 =                          41a20000
answer 10 =                          42c80000
answer 11 =                          40dc8000
answer 12 =                          40440000
answer 13 =                          42704000
answer 14 =                          43f78800
answer 15 =                          42992000
zynq>
```

範例說明 – accelerator講解(1/2)

- 於Design Source下，找到accelerator並點擊開啟
- 硬體實現的加速器下面有規劃一塊 SRAM,是我們用來存取從 ARM 端發送過來的 16 筆數值。透過控制暫存器的控制,我們於第一個周期寫入 SRAM 原始的數值,兩個周期後取出原數值並做平方計算。
- 從外部 start flag 得知何時開始計算, bsy 由cnt控制用於表示正在計算中




```
always @(posedge clk or negedge rst_n)
begin
    if(~rst_n)
        bsy <= 1'b0;
    else if(start == 1'b1 && bsy == 1'b0)
        bsy <= 1'b1;
    else if (cnt == 8'd127 && bsy ==1'b1)
        bsy <= 1'b0;
    else
        bsy <= bsy;
end
always @(posedge clk or negedge rst_n)
begin
    if(~rst_n)
        cnt <= 8'b0;
    else if(bsy == 1'b1 && (cnt <=8'd127))
        cnt <= cnt + 8'b1;
    else
        cnt <= 8'b0;
end
```

accelerator.v

範例說明 - accelerator講解(2/2)

- 控制暫存器的數值範圍決定寫入的記憶體位址與數值

```
assign wen_in = (bsy)?cnt[0]:wen;
assign addr_in = (bsy)?cnt[6:1]:addr[7:2];
assign din_in = (bsy)?mul_output:din;
```

- 利用浮點數乘法器算出 f(x)=x^2 之結果 ,將值傳給mul_output

```
fpu mpy(.clk(clk),.opcode(2'b11), .A(dout), .B(dout), .dout(mul_output));
```

範例說明- C 配置硬體上記憶體位址(1/2)

- 以 C 在 ARM 平台實作 Master, ZYNQ_BASE為記憶體允許存取資料的起始位址
- 因為總共有 16 筆data, 且每筆皆為32bits=4Bytes, 故需要64Bytes的空間
- 其餘作為 start, status signal使用, 相當於上述的control register

```
#define ZYNQ_BASE            0x80000000  //base address for 16 sequence of integer
#define START_RUN_OFFSET     0x00000040  //for start signal
#define STATUS_OFFSET        0x00000044  //for status signal
```

- 變數宣告都使用 volatile 聲明方式宣告，用意是告訴 compiler 不要簡化變數變化的行為。舉例來說，如果變數從0變1又從1變0，compiler可能會自動把這段code省略，但是對於硬體來說，值的變動是有差的

```
volatile int fd = open( "/dev/mem", O_RDWR);//volatile是一個變數聲明限定詞,可能會在任何時刻被意外的更新
volatile int map_len = 0x400;
volatile unsigned int* io = (volatile unsigned int *)mmap(NULL, map_len, PROT_READ | PROT_WRITE, MAP_SHARED, fd, ZYNQ_BASE);
    /*
    NULL :指向欲映射的核心起始位址,NULL代表讓系統自動選定位址
    map_len :代表映射的大小。將文件的多大長度映射到記憶體
    參數prot :映射區域的保護方式。可以為以下幾種方式的組合：
        PROT_EXEC 映射區域可被執行
        PROT_READ 映射區域可被讀取
        PROT_WRITE 映射區域可被寫入
        PROT_NONE 映射區域不能存取
    參數flags :影響映射區域的各種特性。在調用mmap()時必須要指定MAP_SHARED 或MAP_PRIVATE。
        MAP_SHARED 允許其他映射該文件的行程共享，對映射區域的寫入數據會複製回文件。
        MAP_PRIVATE 不允許其他映射該文件的行程共享，對映射區域的寫入操作會產生一個映射的複製(copy-on-write)，對此區域所做的修改不會寫回原文件。
    fd :要映射到核心中的文件
    ZYNQ_BASE :從文件映射開始處的偏移量
    */
```

範例說明- C 配置硬體上記憶體位址(2/2)

- 將準備計算的值寫入陣列(參照.excel檔), 再寫入映射的虛擬記憶體位址

```
m[0] = 0x3f000000;//0.5
m[1] = 0x3e99999a;//0.3
m[2] = 0x3e800000;//0.25
m[3] = 0x3f4ccccd;//0.8
m[4] = 0x3f19999a;//0.6
m[5] = 0x3fc00000;//1.5
m[6] = 0x3fa00000;//1.25
m[7] = 0x3f400000;//0.75
m[8] = 0x40600000;//3.5
m[9] = 0x40900000;//4.5
m[10] = 0x41200000;//10
m[11] = 0x40280000;//2.625
m[12] = 0x3fe00000;//1.75
m[13] = 0x40f80000;//7.75
m[14] = 0x41b20000;//22.25
m[15] = 0x410c0000;//1.75
```

```
printf("// ------------
for (i=0;i<16;i++)
    *(io+i) = m[i];
```

作業及評分方式

- 作業內容：修改範例, 完成 f(x)=x^3
- 作業測資與結果：請依照excel檔內容,使用要求的測資並顯示給予的範例答案
- 評分方式：課程範例 50%

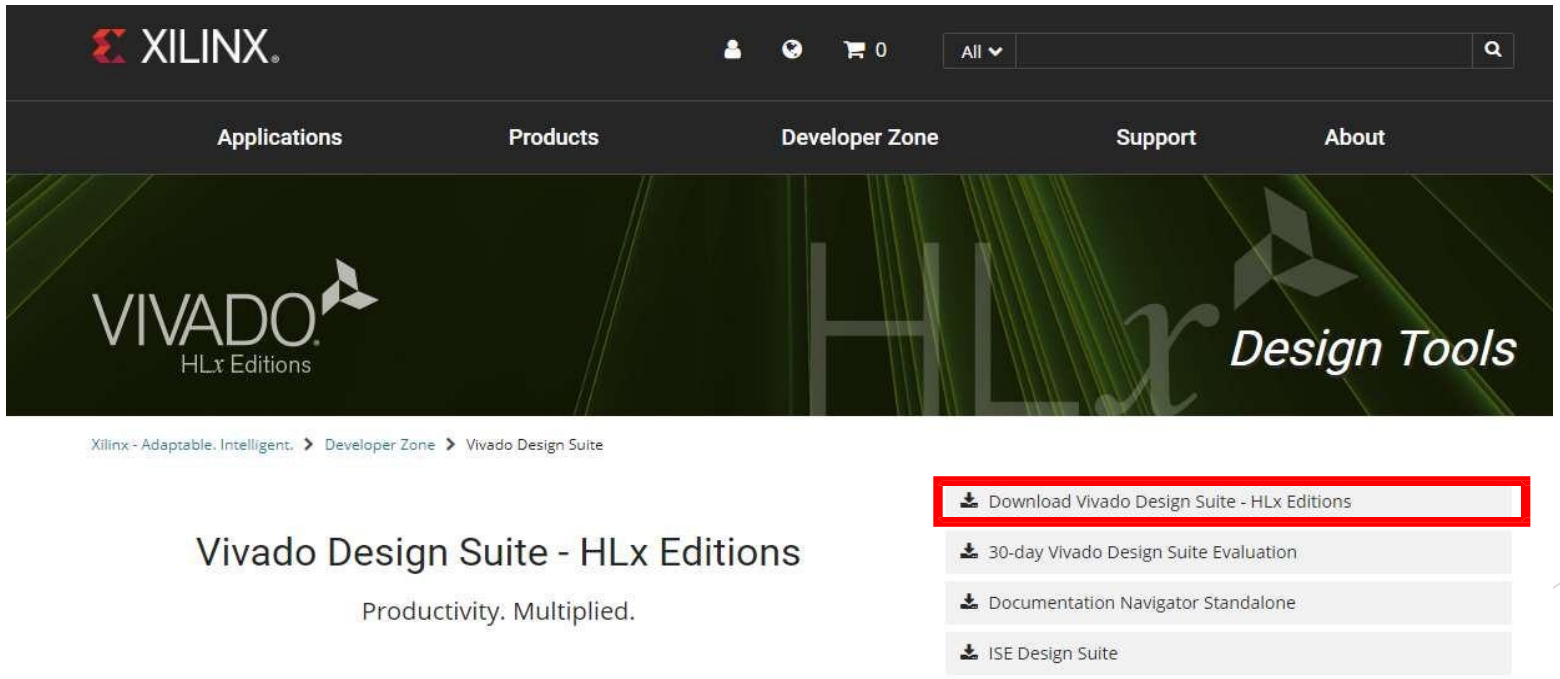
  作業練習 30%

  隨堂練習 20%

  影片有較多說明：https://youtu.be/jAMPmlByvbM
- Demo 地點:工程一館206

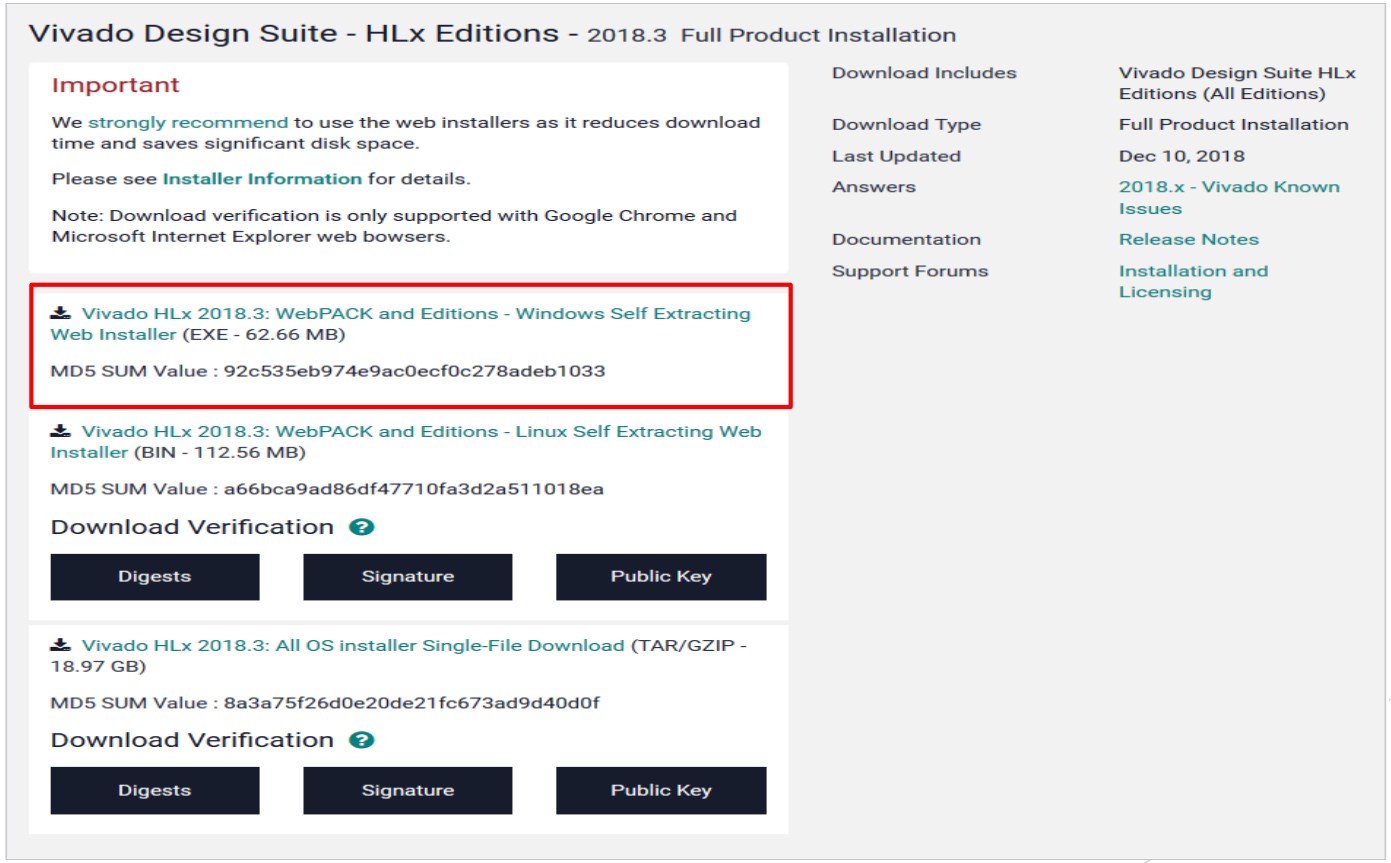
附錄 - Vivado 安裝(1/8)

- 前往Vivado官網, 下載Vivado Design Suite – HLx Editions



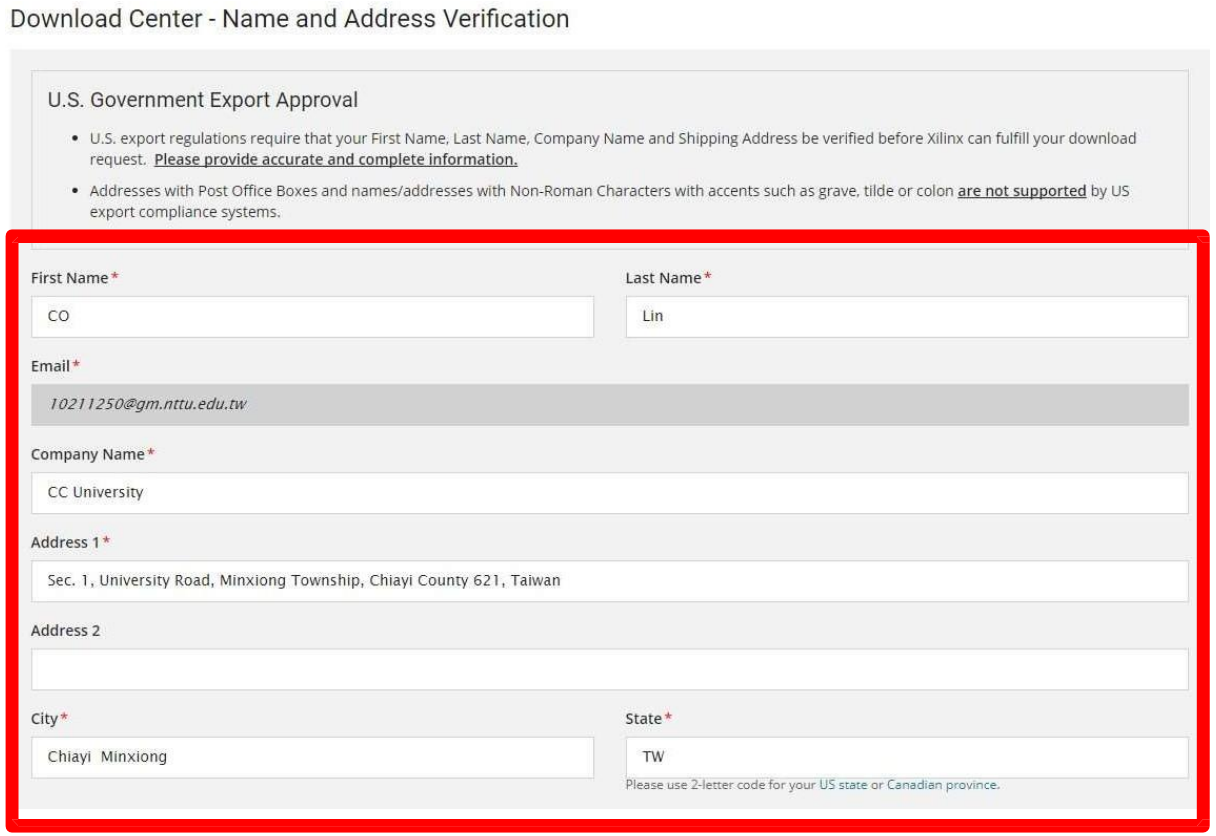
附錄 - Vivado 安裝(2/8)

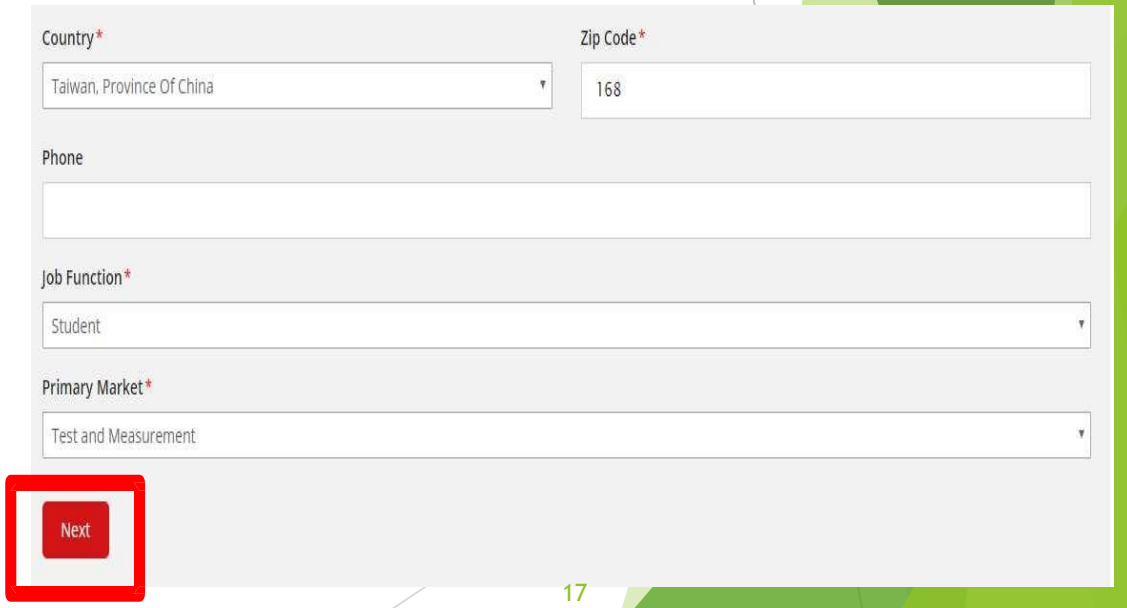
- 點選Vivado HLx 2018.3:WebPACK and Editions



附錄 - Vivado 安裝(3/8)

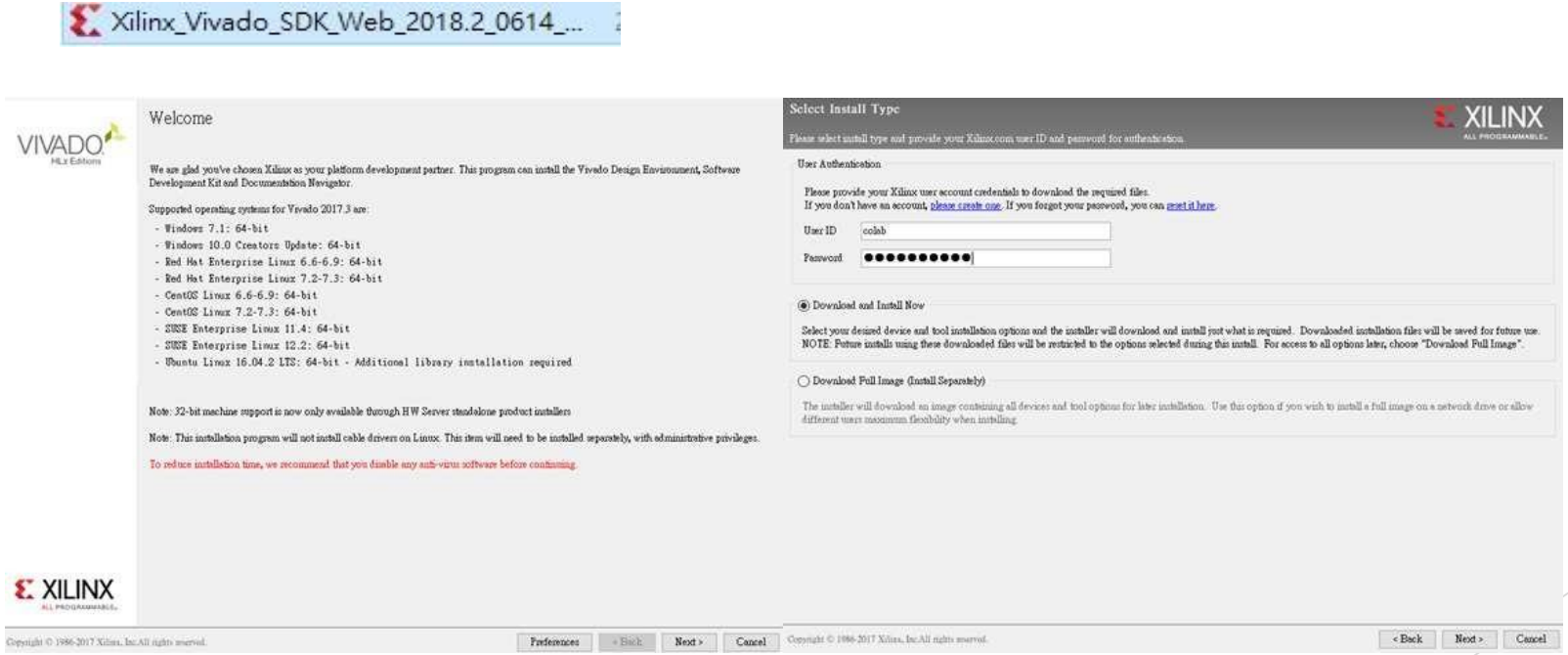
- 完善個人資料後下載**(此範例不需要修改)**





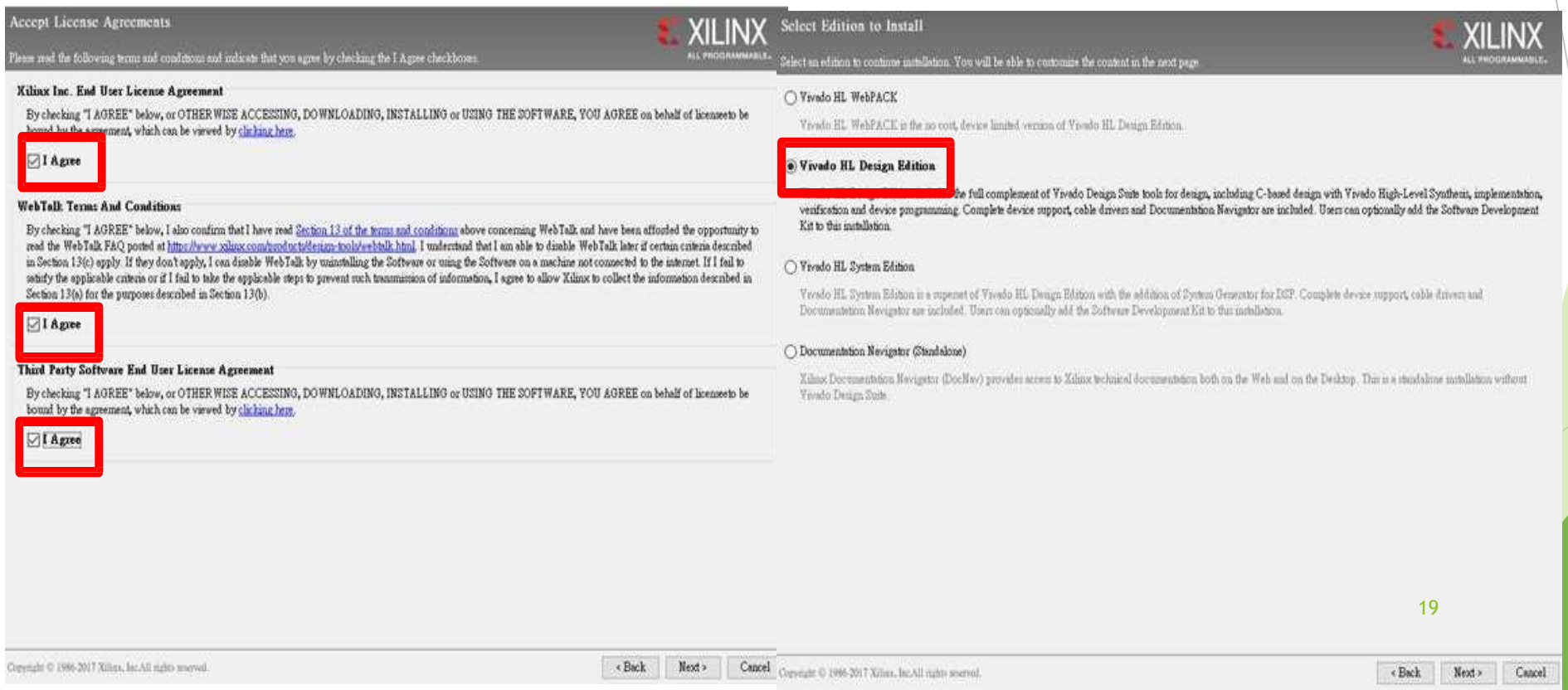
附錄 - Vivado 安裝(4/8)

- User ID：colab　　Password：cocococo4*
- **備註：請大家不要修改帳號相關資料**



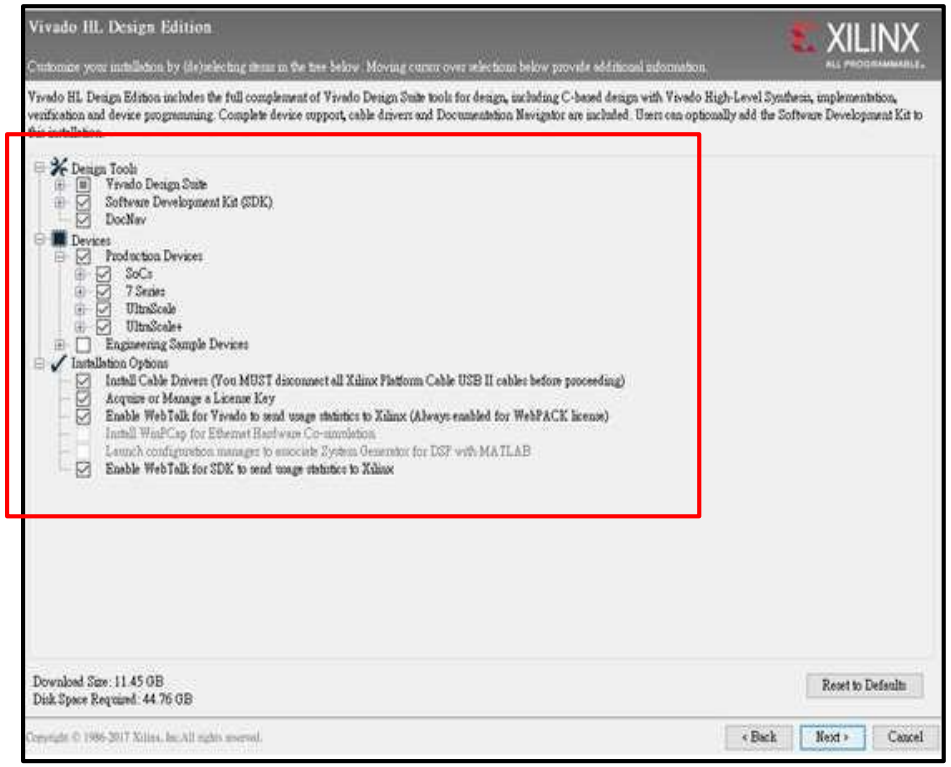
附錄 - Vivado 安裝(5/8)

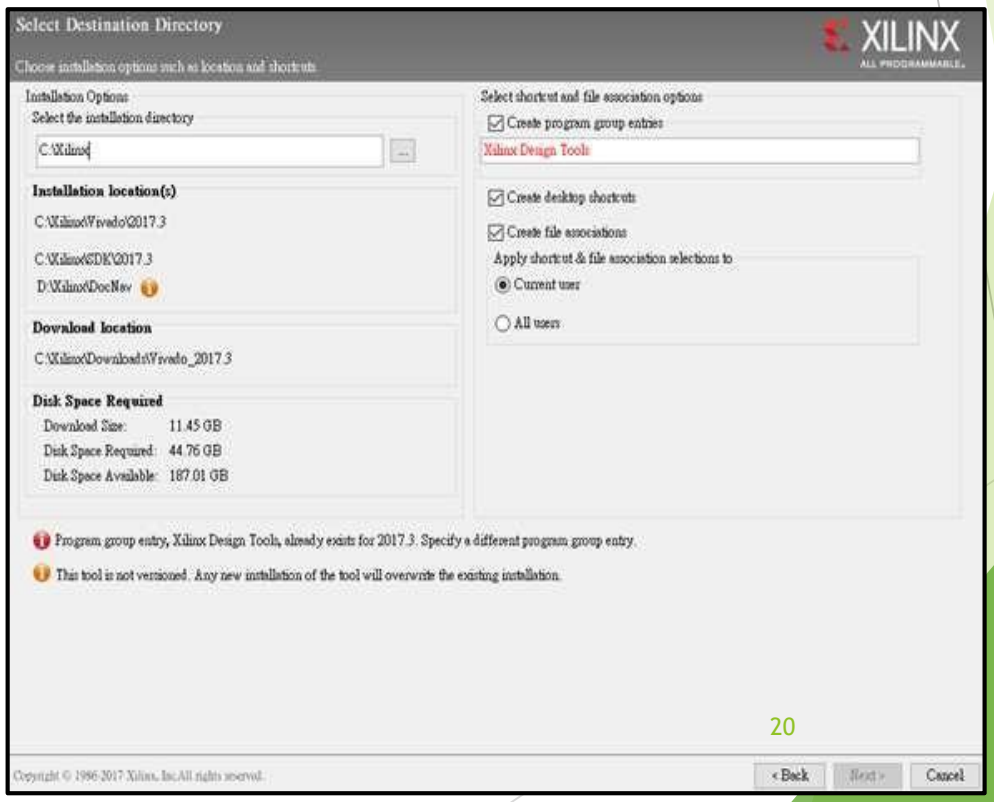
- 開始安裝



附錄 - Vivado 安裝(6/8)

- 開始安裝



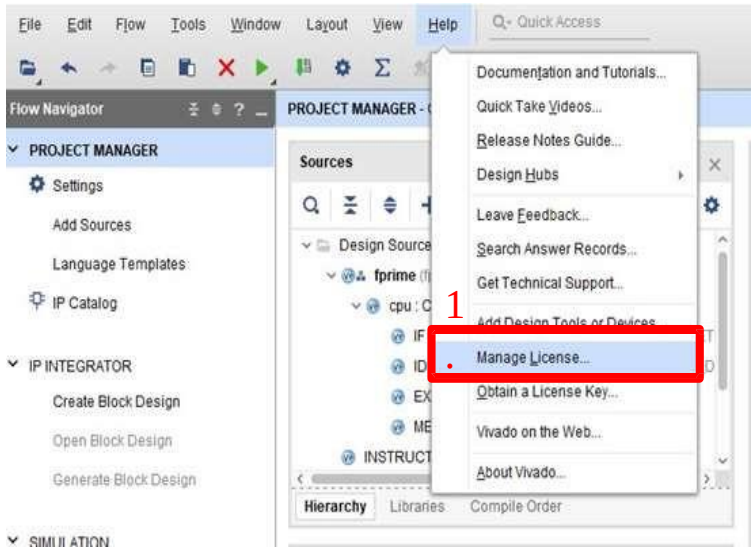


附錄 - Vivado 安裝(7/8)

- 等待安裝 (需要一點時間)

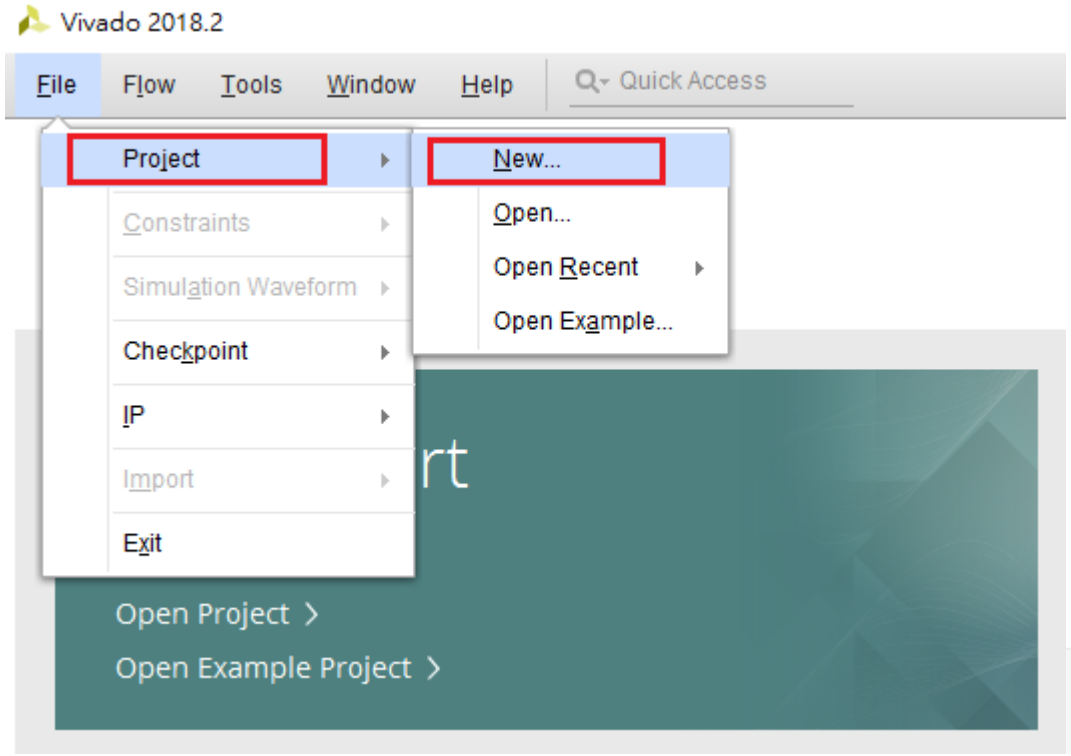
附錄 - Vivado 安裝(8/8)

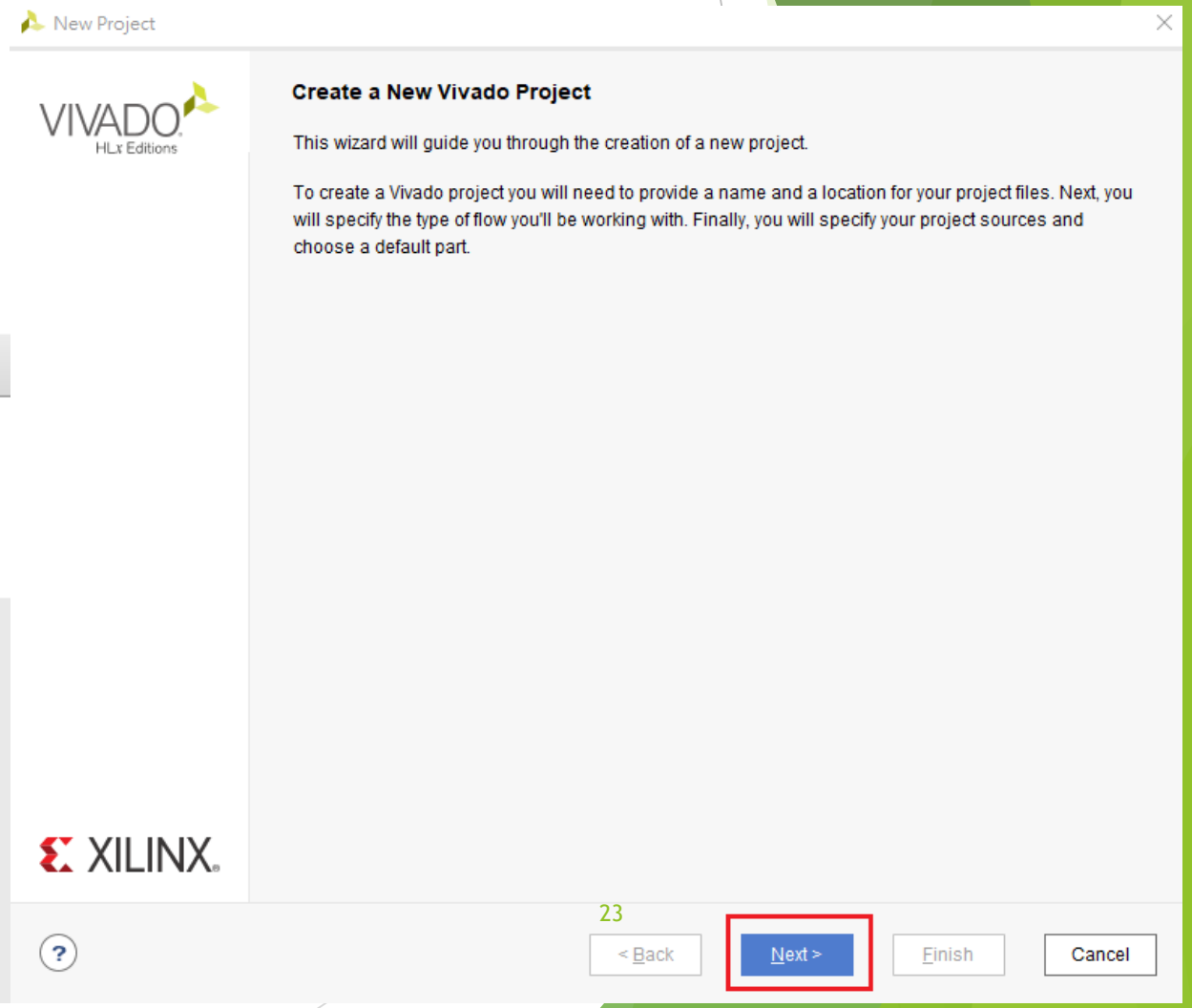
- License：https://drive.google.com/file/d/1ddZw3QlqKx_4s5ate7_dDbikZX7GN5h-/view?usp=sharing
- 加入License



附錄 - 創建新檔(1/4)

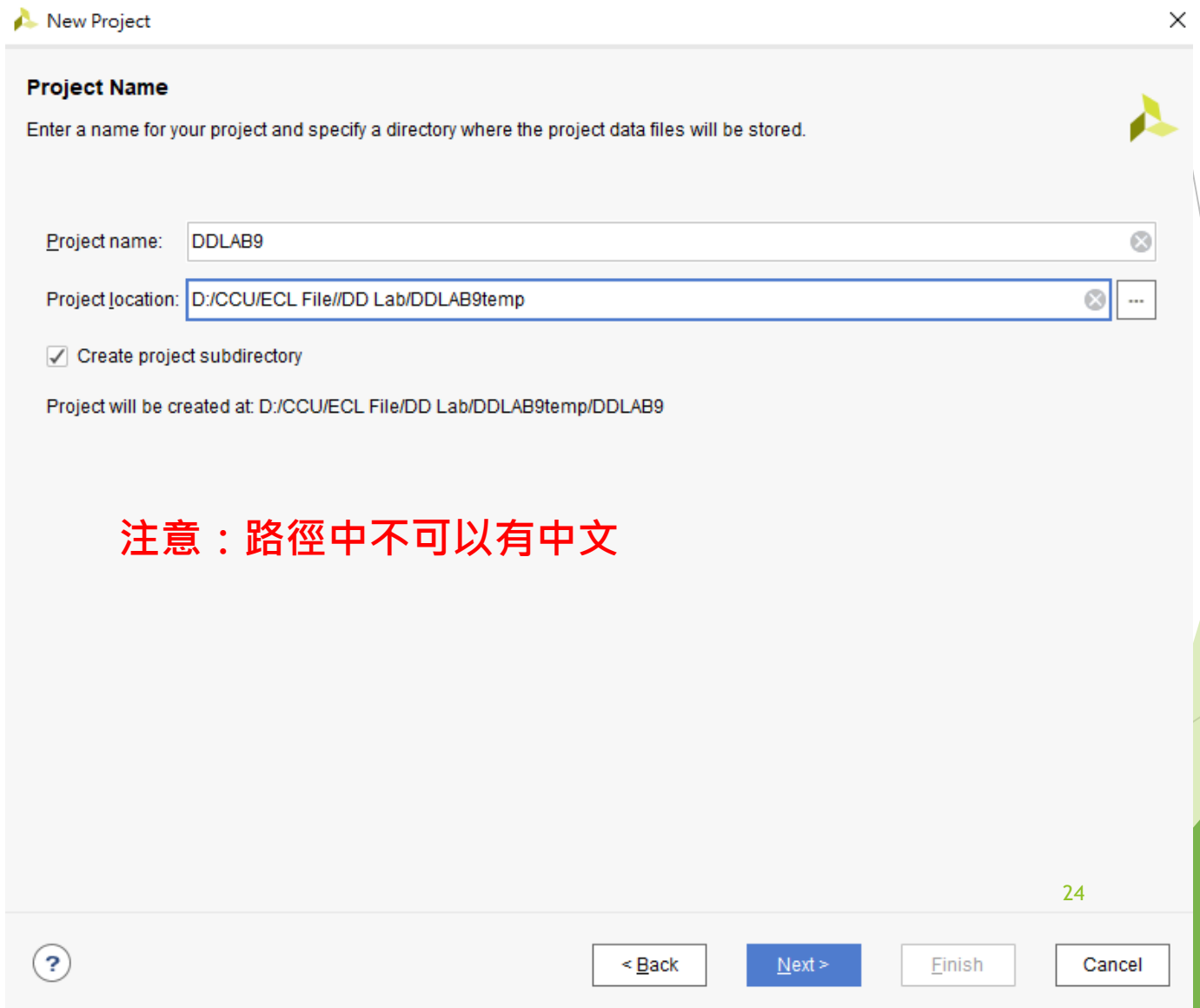
- 開啟Vivado 並新增專案



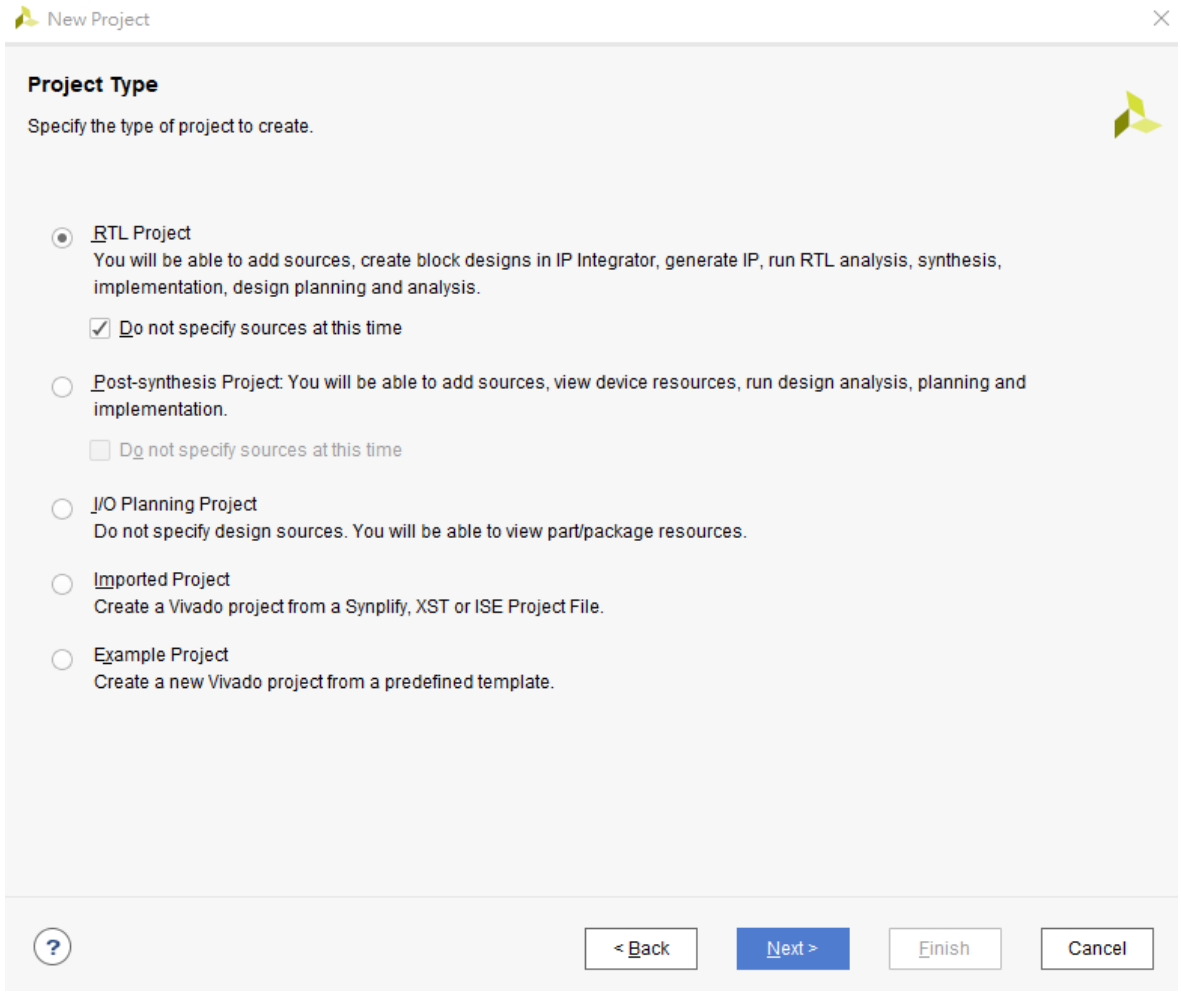


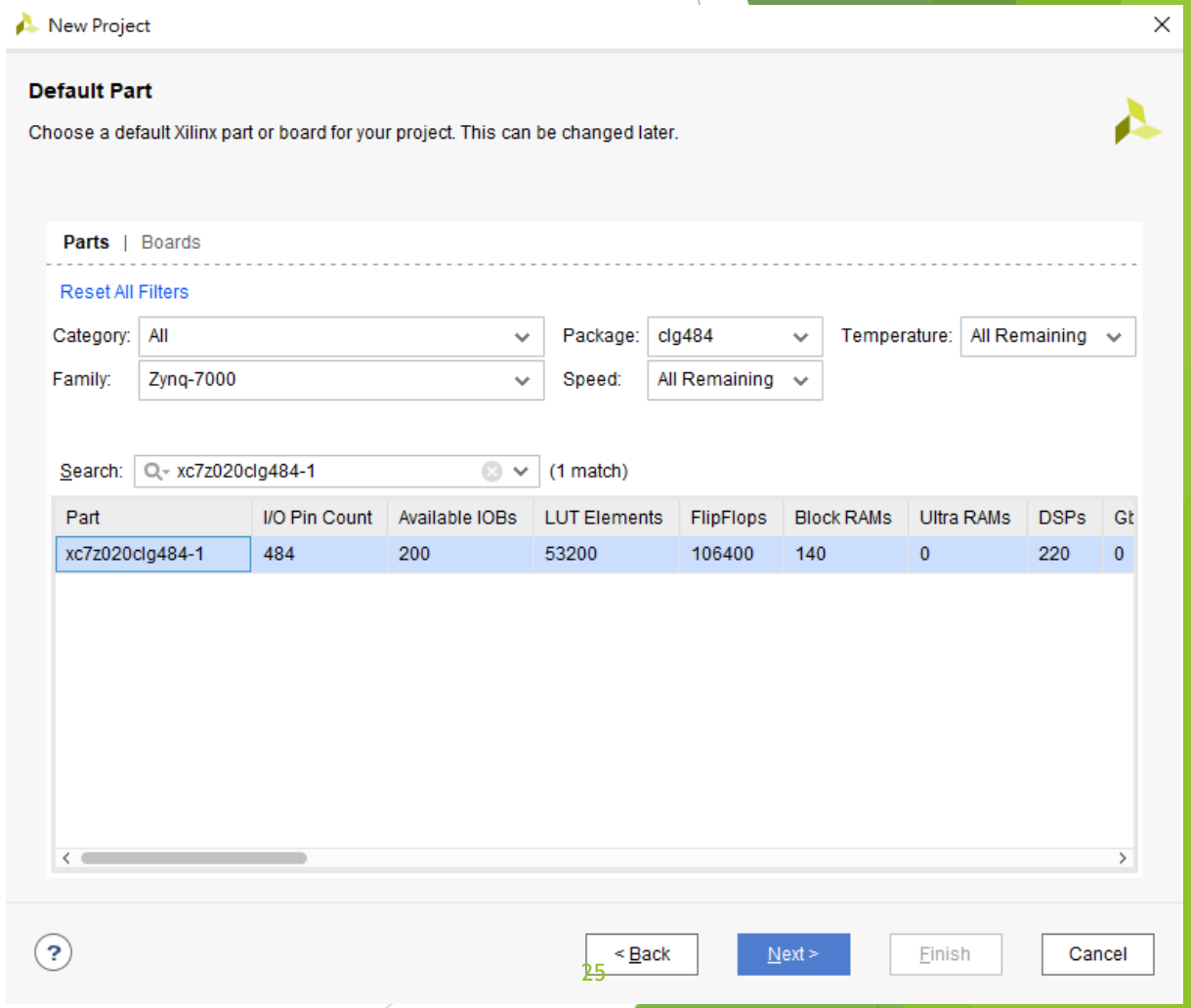
附錄 - 創建新檔(2/4)

- 專案命名

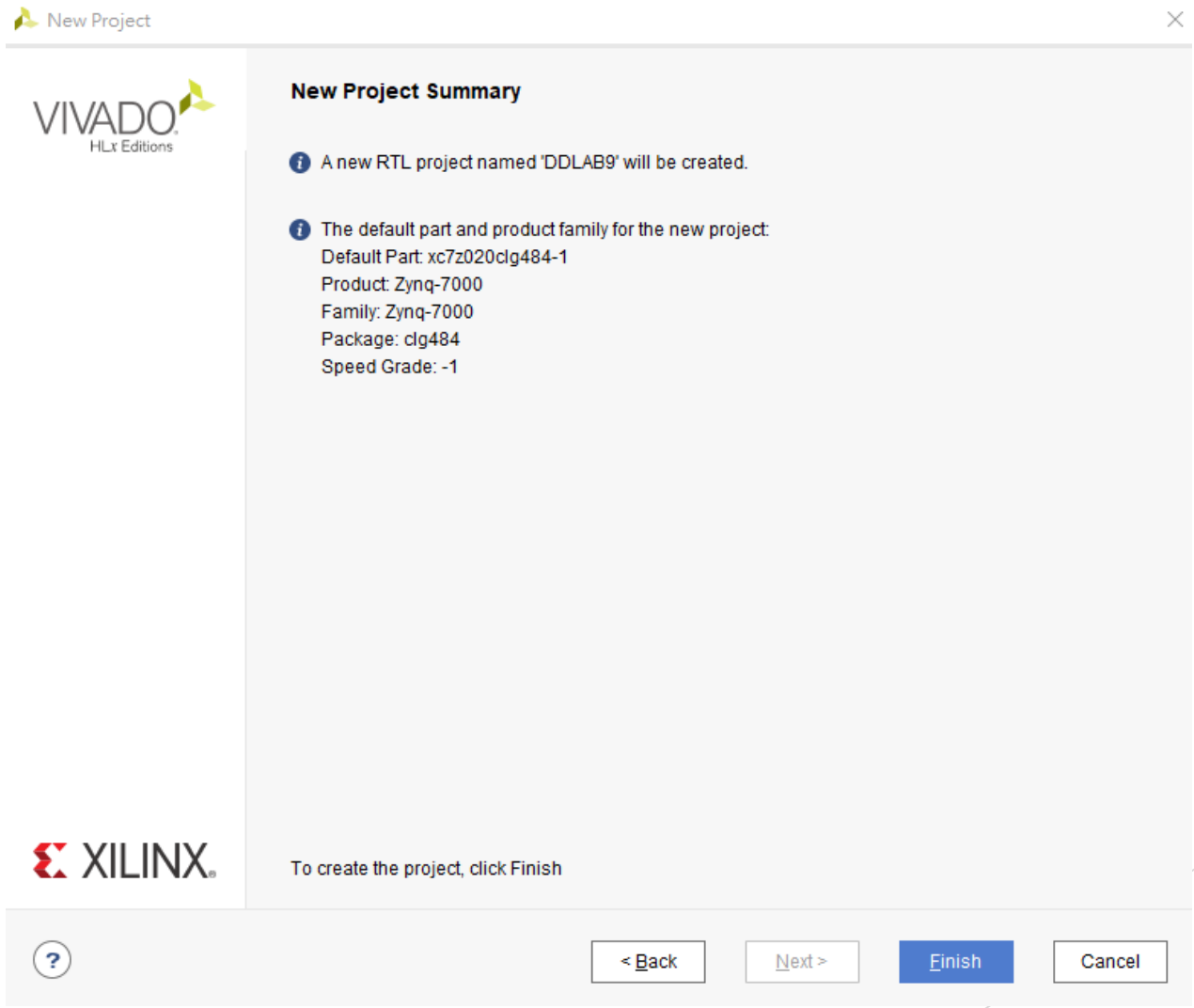


附錄 - 創建新檔(3/4)


New Project
Project Type
Specify the type of project to create.
RTL Project
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
Do not specify sources at this time
Post-synthesis Project: You will be able to add sources, view device resources, run design analysis, planning and implementation.
Do not specify sources at this time
I/O Planning Project
Do not specify design sources. You will be able to view part/package resources.
Imported Project
Create a Vivado project from a Synplify, XST or ISE Project File.
Example Project
Create a new Vivado project from a predefined template.
< Back
Next >
Finish
Cancel


New Project
Default Part
Choose a default Xilinx part or board for your project. This can be changed later.
Parts | Boards
Reset All Filters
Category: All
Package: clg484
Temperature: All Remaining
Family: Zynq-7000
Speed: All Remaining
Search: xc7z020clg484-1
(1 match)
Part
I/O Pin Count
Available IOBs
LUT Elements
FlipFlops
Block RAMs
Ultra RAMs
DSPs
xc7z020clg484-1
484
200
53200
106400
140
0
220
0
< Back
Next >
Finish
Cancel

附錄 - 創建新檔(4/4)


New Project
VIVADO
HLx Editions
New Project Summary
A new RTL project named 'DDLAB9' will be created.
The default part and product family for the new project:
Default Part: xc7z020clg484-1
Product: Zynq-7000
Family: Zynq-7000
Package: clg484
Speed Grade: -1
XILINX
To create the project, click Finish
?
< Back
Next >
Finish
Cancel