

Jini Intelligent Computing Workbook of Lab. #3

Preamble

Lab. #3在AlveoU50的Platform專案目錄下有四個Applications專案目錄：

- vts_Opt1Baseline
- vts_Opt2KernelParallel
- vts_Opt3DataBurst
- vts_Opt4ArrayPartition

以上目錄中皆包含該專案的原始碼檔。

1. Introduction

本實驗為Vitis OpenCL/XRT實作，以Xilinx Alveo U50 PCIe加速卡為基礎。Xilinx Alveo U50為PCIe介面之FPGA加速卡，以Linux server為平台透過建置Xilinx XRT runtime架構，再以OpenCL語言開發host program，將bitstream(.xclbin)檔案下載至Xilinx Alveo U50加速卡，並運行host program的流程控制。

此外本實驗就算沒有Alveo U50加速卡，仍然可以在使用者PC做到Software/Hardware Emulation，Hardware Emulation的模擬結果與真實在FPGA運行相近。

Note：

因Windows版本的Vitis不支援Alveo U50等PCIe介面之FPGA加速卡，本次實驗將全部在Linux系統上實作，若無Linux PC亦可在Windows上以Oracle VM VirtualBox等virtual machine運行。採用VM請特別注意分配給VM的記憶體不要太少，建議分配8 GB以上，CPU也可以多分配一些以加快模擬速度。

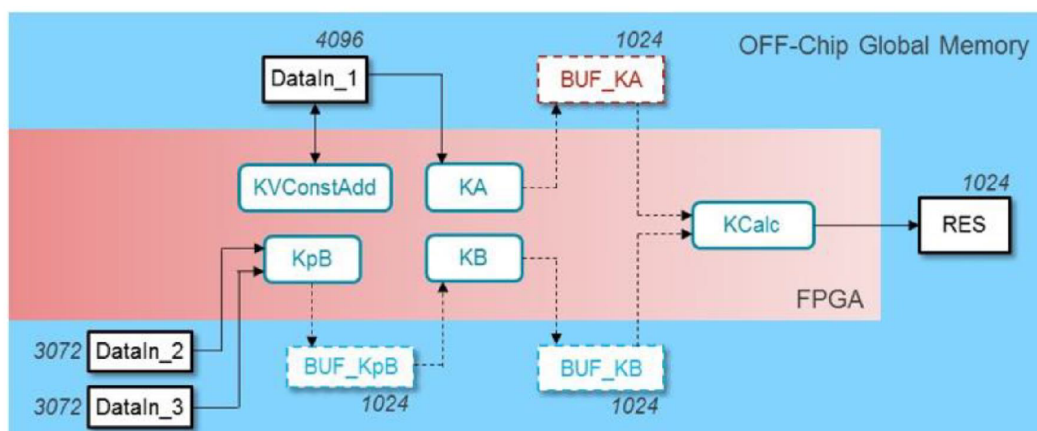
2. Vitis Application Acceleration

【施作環境為在使用者PC/laptop/notebook (Linux Base)。】

本實驗共有四個專案，分別對應Baseline、Kernel Parallel、Data Burst及Array Partition四種不同的組態。

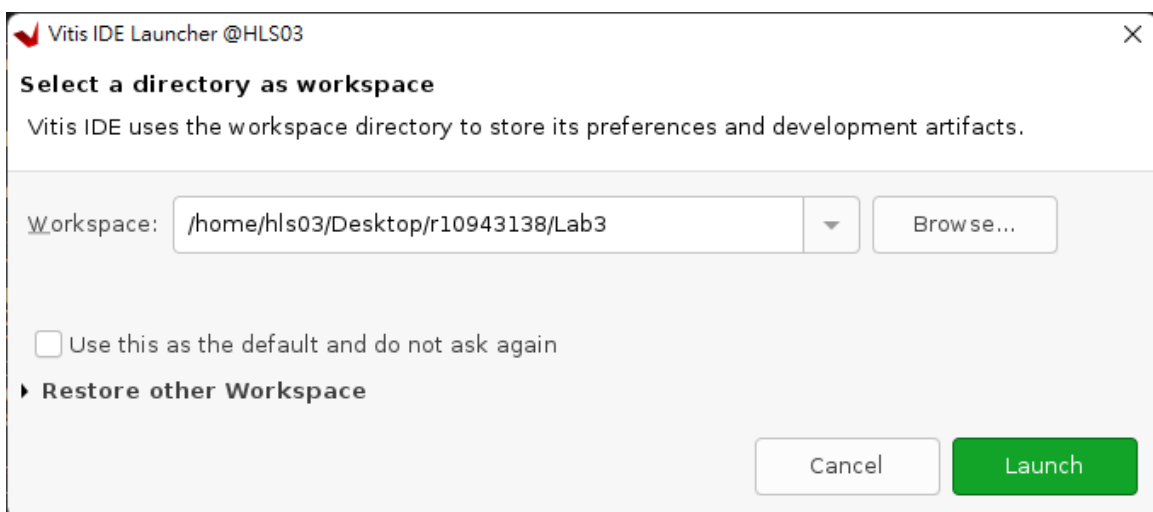
四個組態的實驗步驟皆相同，請仔細比較各組態source code及產生的Application Timeline與Profile Summary的差異。下方步驟以Baseline作為範例。

下圖為本次實驗的架構，由五個kernel function及七個在global memory中的data buffer組成。

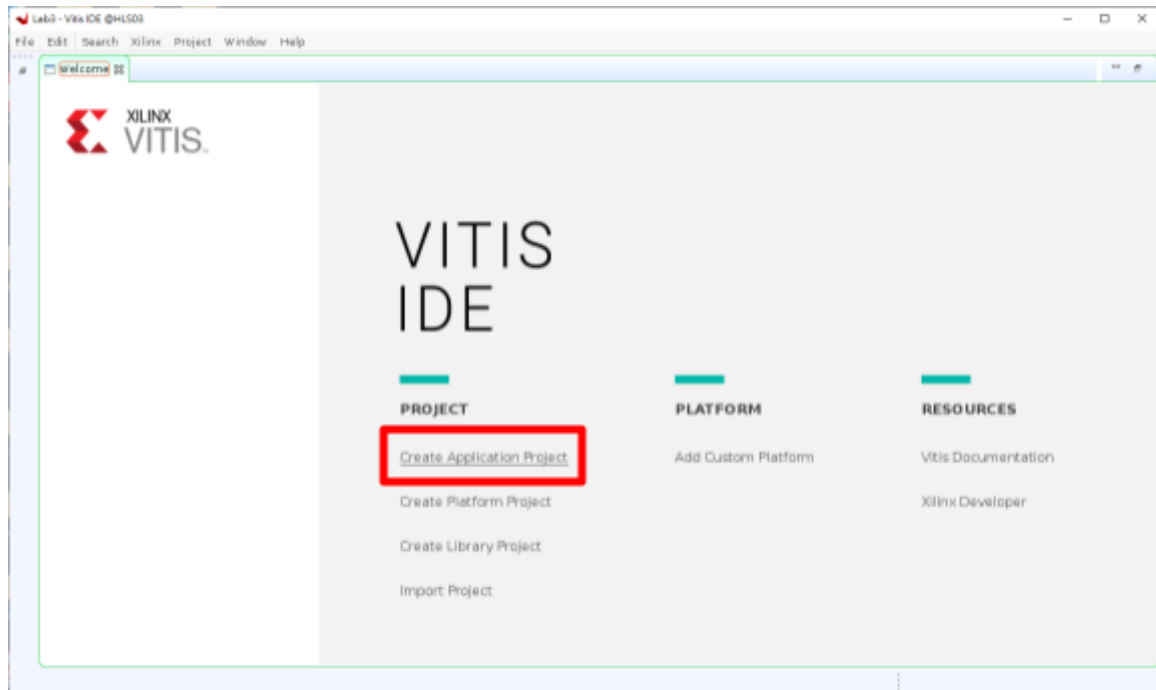


2.1. Create and Setup Project

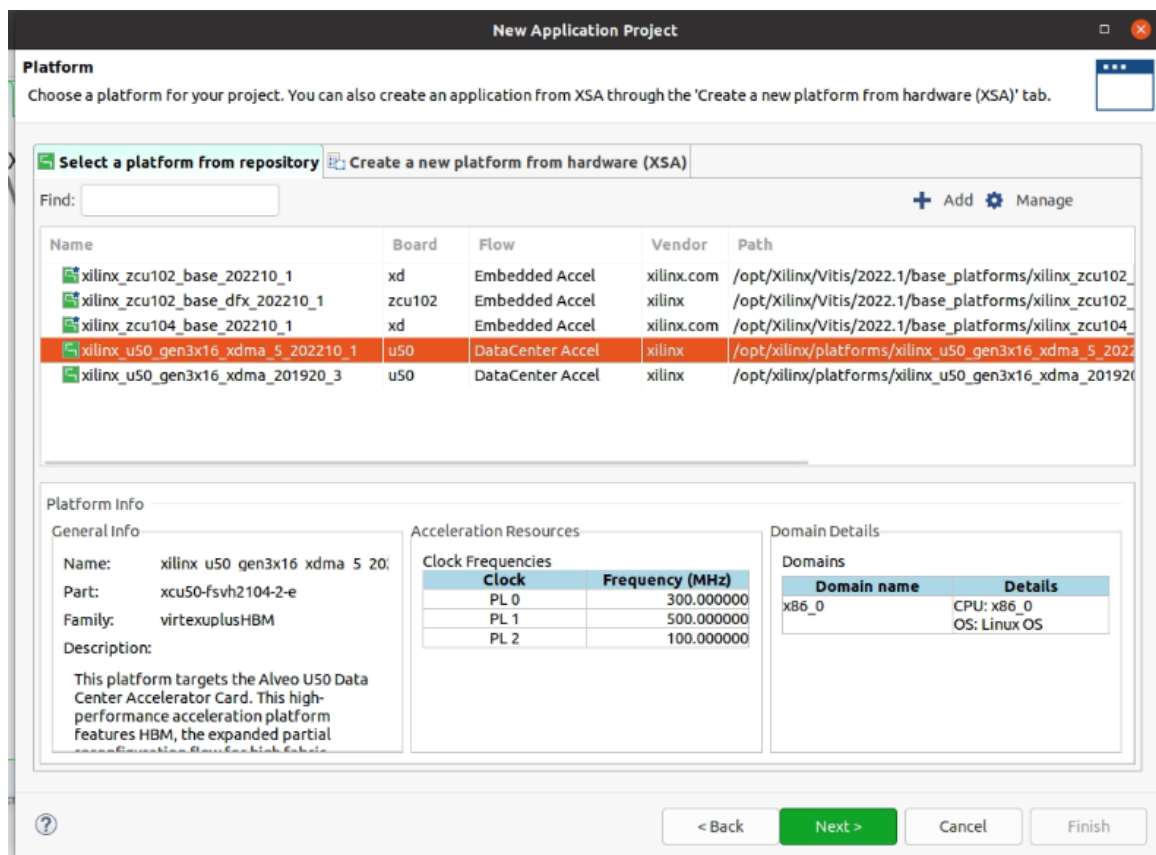
登入先前註冊好的帳號，在Terminal輸入vitis開啟Vitis程式，並設定好運行的Launch directory。



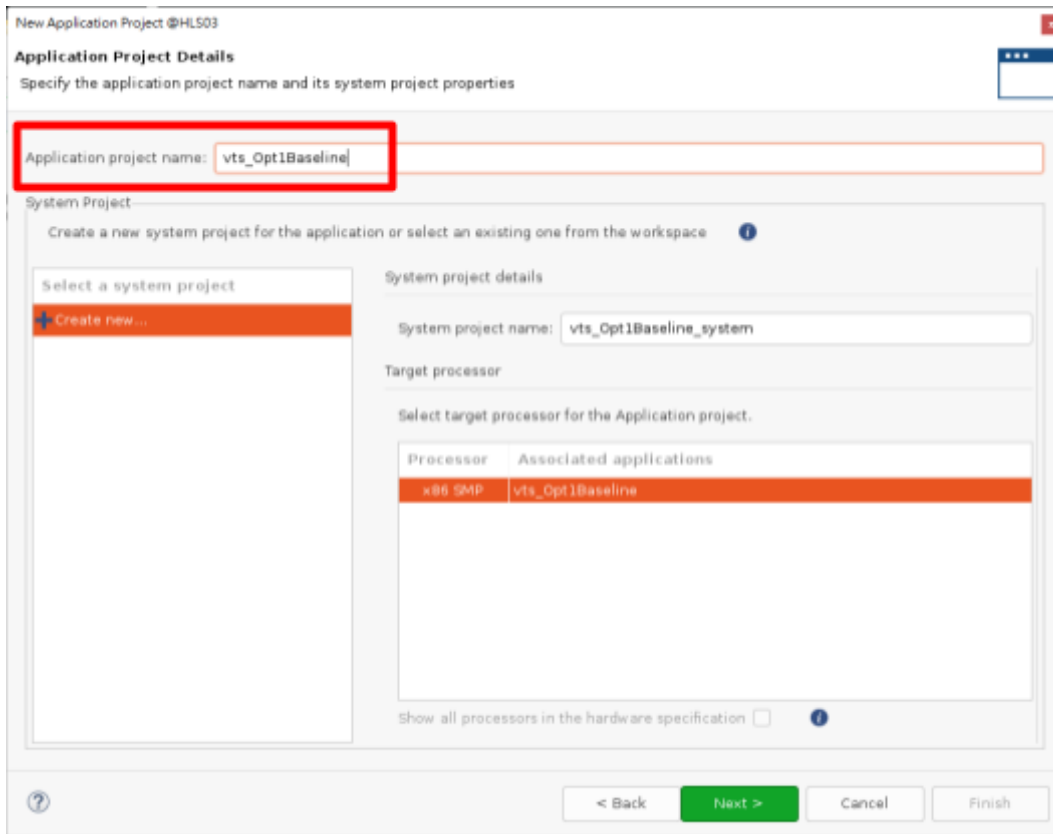
開啟主畫面後，點選Create Application Project。



Platform選擇xilinx_u50_gen3x16_xdma_5_202210_1 (注意是選2022的喔)



為專案命名，下方System project name會自動填上毋須修改。



New Application Project @HLS03

Application Project Details

Specify the application project name and its system project properties

Application project name:

System Project

Create a new system project for the application or select an existing one from the workspace

Select a system project

- Create new...

System project details

System project name:

Target processor

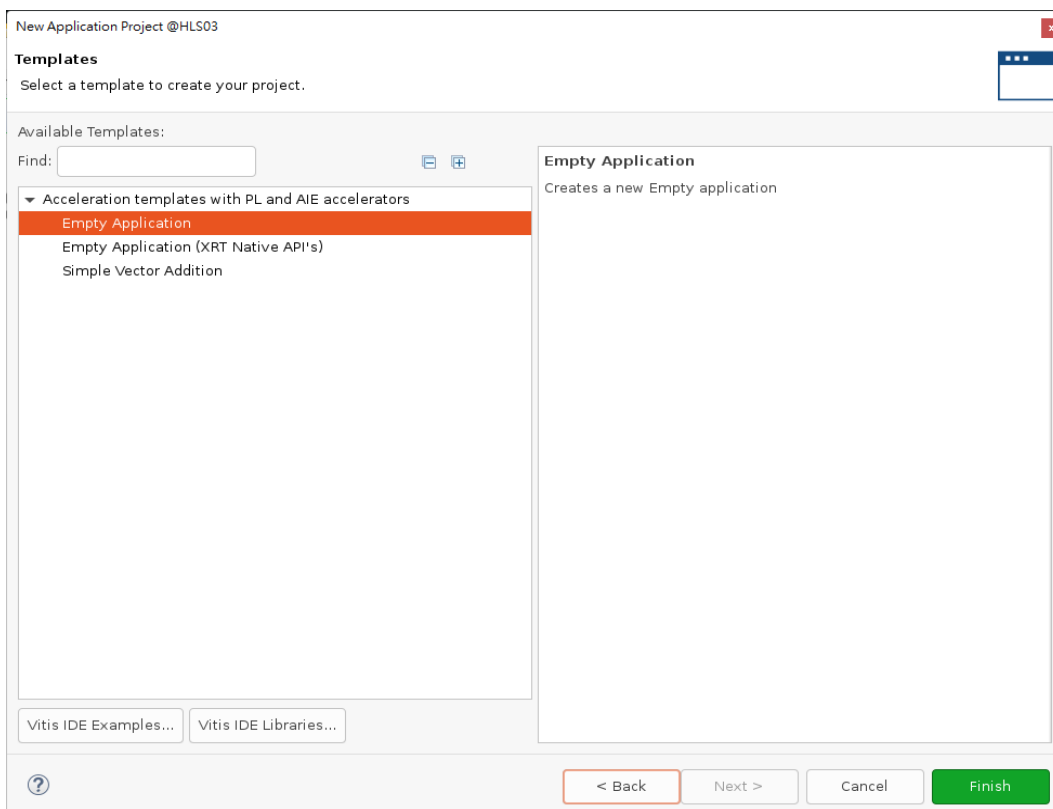
Select target processor for the Application project.

Processor	Associated applications
x86 SMP	vts_Opt1Baseline

Show all processors in the hardware specification ☐

< Back Next > Cancel Finish

選擇Empty Application，點選Finish建立專案。



New Application Project @HLS03

Templates

Select a template to create your project.

Available Templates:

Find:

▼ Acceleration templates with PL and AIE accelerators

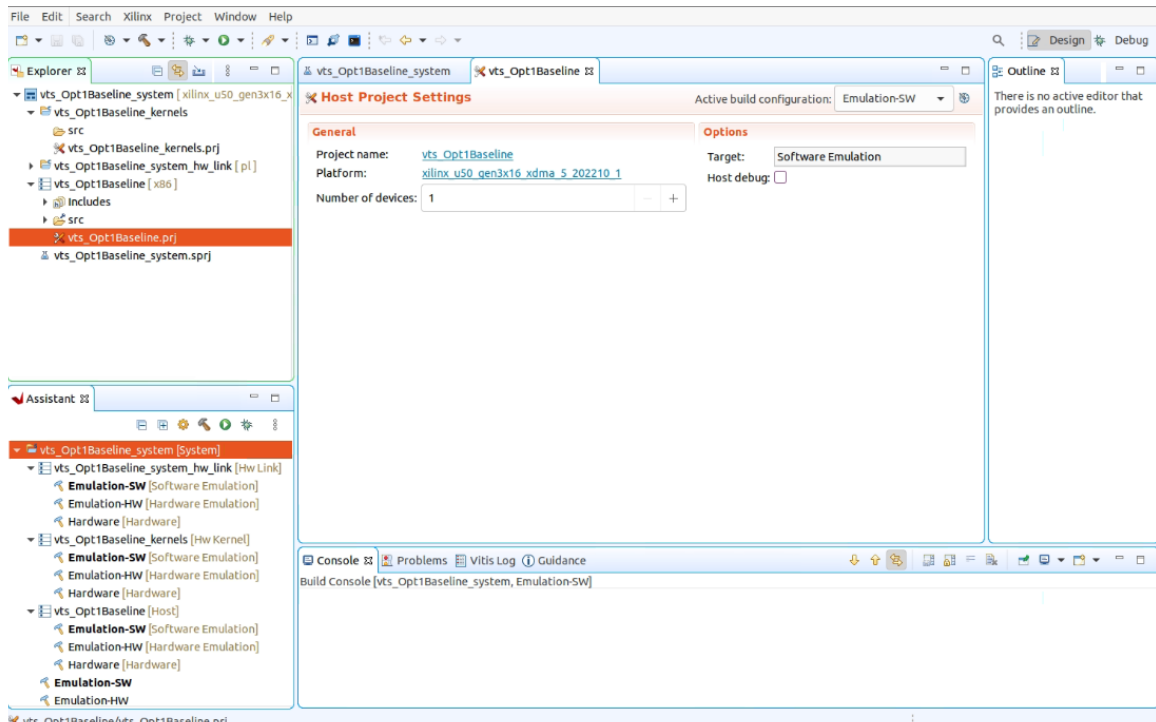
- Empty Application
- Empty Application (XRT Native API's)
- Simple Vector Addition

Vitis IDE Examples... Vitis IDE Libraries...

Empty Application

Creates a new Empty application

< Back Next > Cancel Finish



左上方Explorer內可看到在vts_Opt1Baseline_system專案底下有三個專案：

1. vts_Opt1Baseline_kernels專案負責compile kernel function。
2. vts_Opt1Baseline_system_hw_link專案負責將kernel link起來產生bitstream file (.xclbin)。
3. vts_Opt1Baseline專案負責host program的部分。

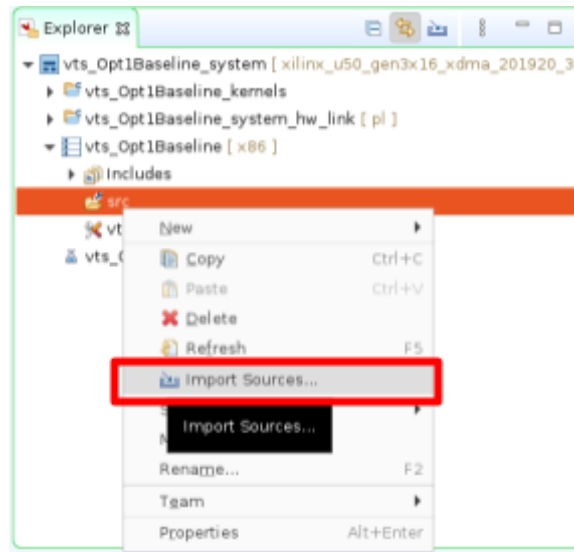
左下方Assistant內顯示了各個專案的建置和模擬狀態，以及各項工作產生的report。

中間的Project Editor顯示專案部分屬性，且可以直接對各專案進行設定。

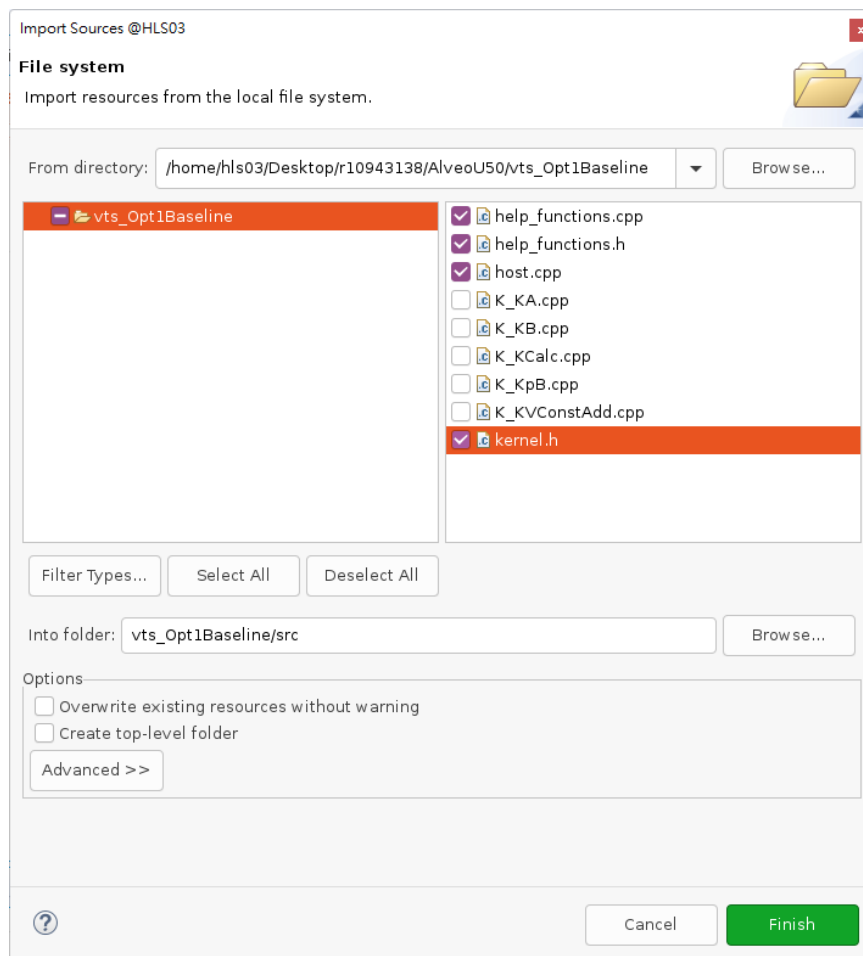
下方有Console顯示工作狀態，且可以在各專案不同組態的console間切換。

建立好專案後第一步要在專案裡加入source code。

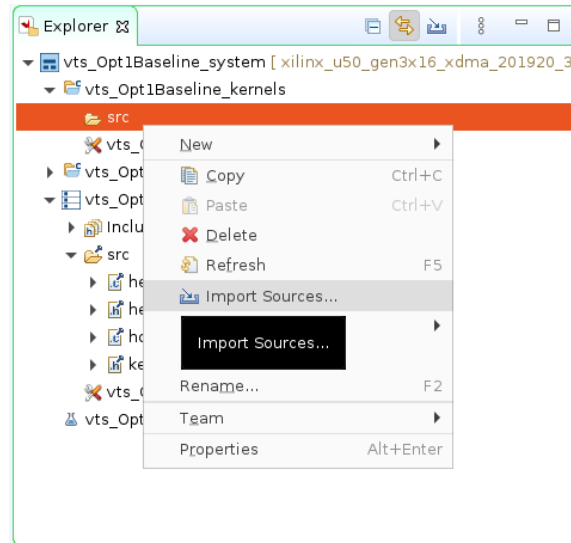
首先在Explorer中右鍵點選vts_Opt1Baseline專案底下的src資料夾，點選Import Sources加入host program的source code。



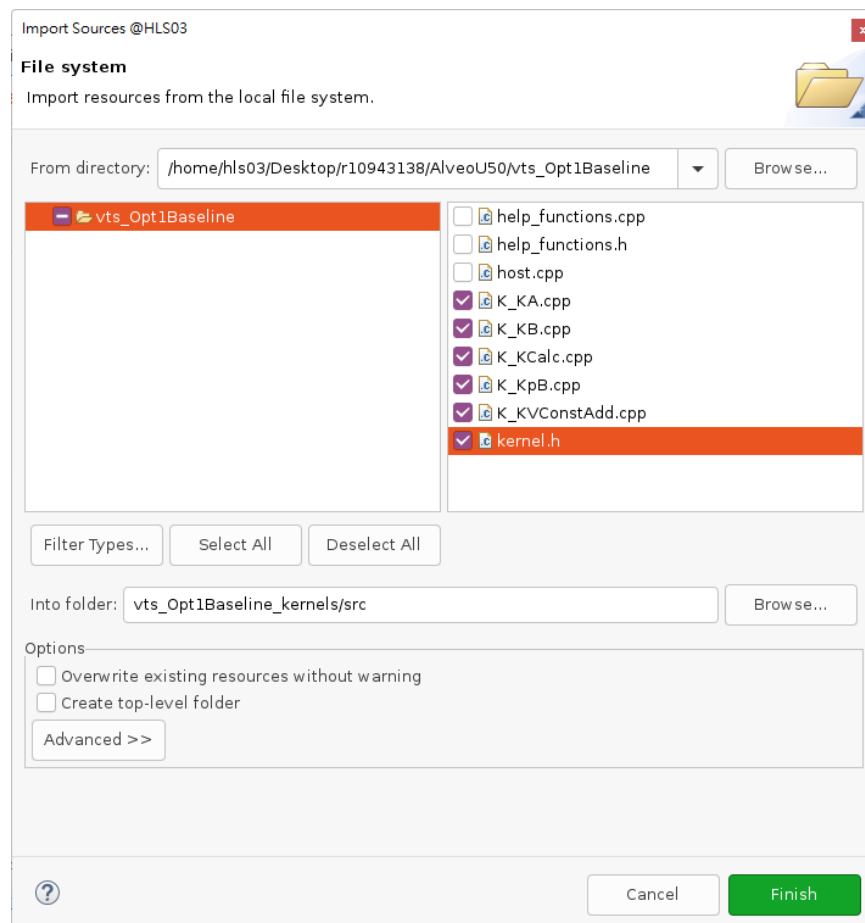
選擇提供的source code資料夾，勾選help_functions.cpp、help_functions.h、host.cpp以及kernel.h。



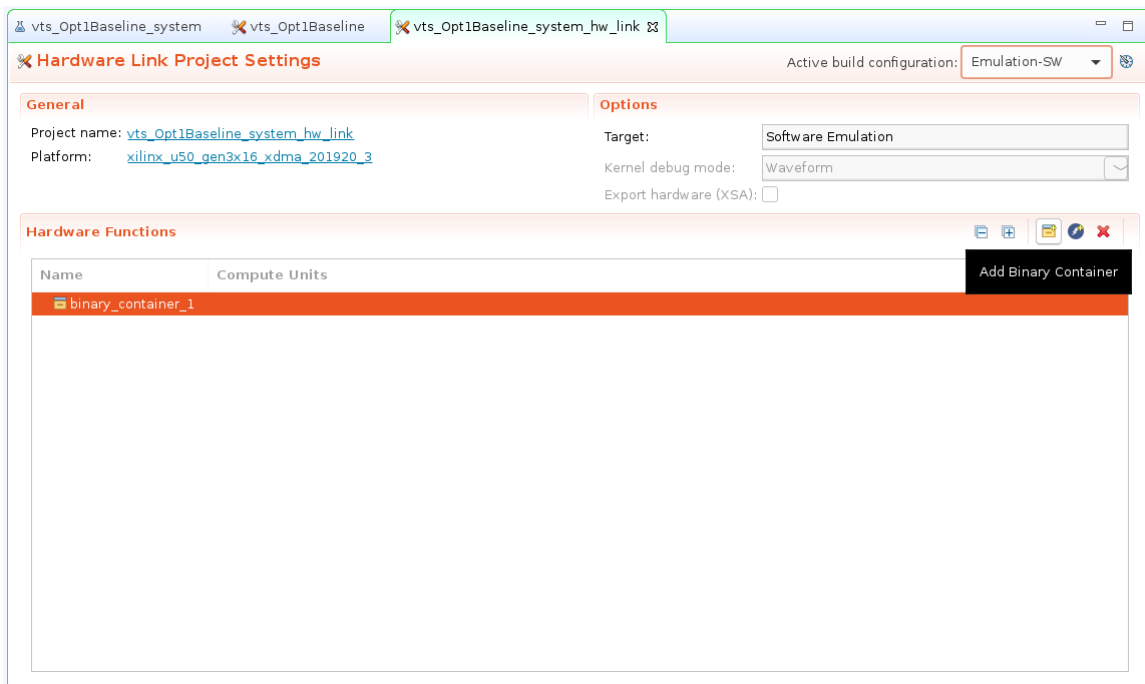
接著在vts_Opt1Baseline_kernels專案底下加入kernel function的source code。



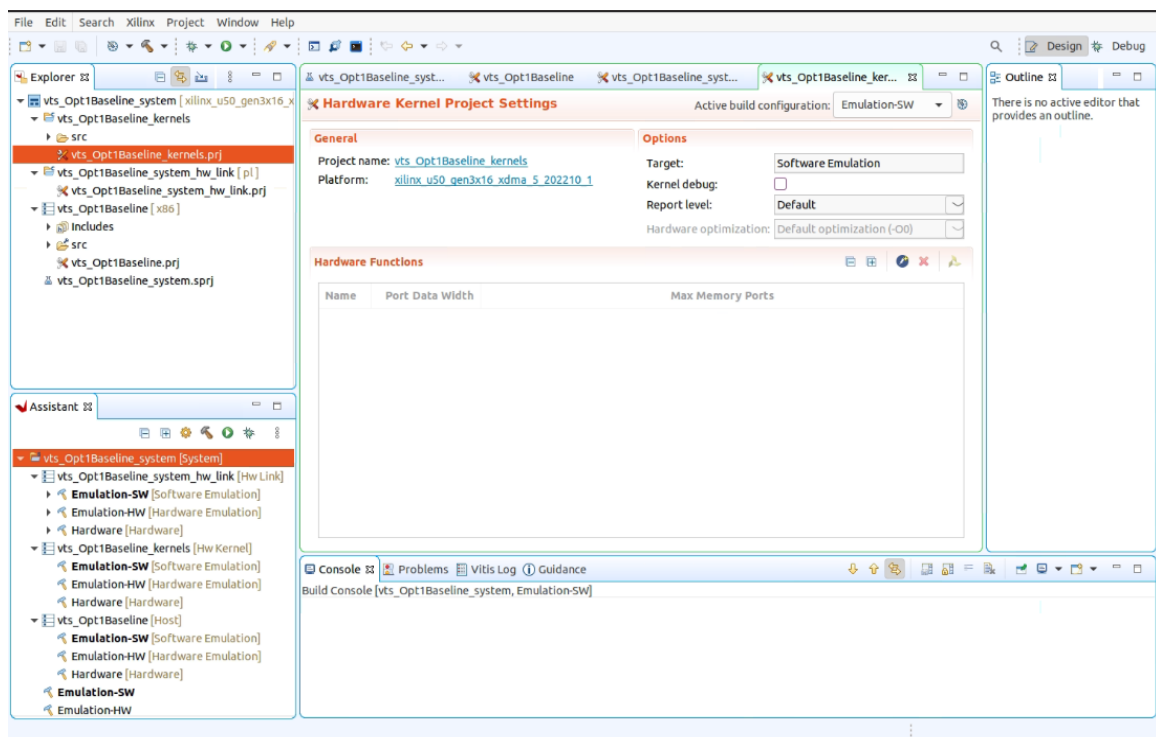
選擇提供的source code資料夾，勾選K_KA.cpp、K_KB.cpp、K_Kcalc.cpp、K_KpB.cpp、K_KVConstAdd.cpp以及kernel.h。



下一步要加入binary container, 打開vts_Opt1Baseline_system_hw_link專案(雙擊vts_Opt1Baseline_system_hw_link.prj), 在Project Editor點選Add Binary Container後, 底下會出現binary_container_1。



最後要加入hardware functions, 打開vts_Opt1Baseline_kernels專案 (vts_Opt1Baseline_kernels.prj), 在Project Editor點選Add Hardware Functions, 並在彈出的視窗中選擇KA、KB、KCalc、KVConstAdd以及KpB加入。完成後底下會列出所有kernel functions。



Add Hardware Functions @HLS03

Select an item to open (? = any character, * = any string):

More Options

Matching items:

- KA(int *, int *) - K_KA.cpp
- KB(int *, int *) - K_KB.cpp
- KCalc(int *, int *, int *) - K_KCalc.cpp
- KVConstAdd(unsigned int, int *) - K_KVConstAdd.cpp
- KpB(int *, int *, int *) - K_KpB.cpp

Cancel
OK

vts_Opt1Baseline_system
vts_Opt1Baseline
vts_Opt1Baseline_system_hw_link
vts_Opt1Baseline_kernels

Hardware Kernel Project Settings
Active build configuration: Emulation-SW

General
Options

Project name: vts_Opt1Baseline_kernels
Platform: xilinx_u50_gen3x16_xdma_201920_3
Target: Software Emulation
Kernel debug:
Report level: Default
Hardware optimization: Default optimization (-O0)

Hardware Functions

Name	Port Data Width	Max Memory Ports
KA	Auto	
KB	Auto	
KCalc	Auto	
KVConstAdd	Auto	
KpB	Auto	

Add Hardware Function...

2.2. Software Emulation

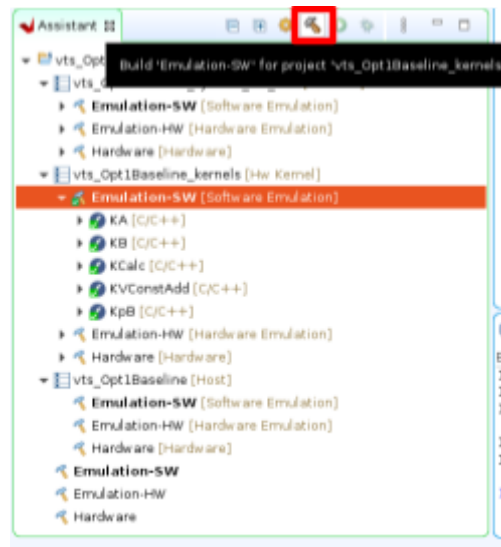
Software emulation是以軟體函式形式直接傳遞引數來模擬結果，類似於Lab. #1 中的C simulation。

2.2.1. Build Project

要執行emulation前要先建置專案，產生模擬需要的執行檔及bitstream file。

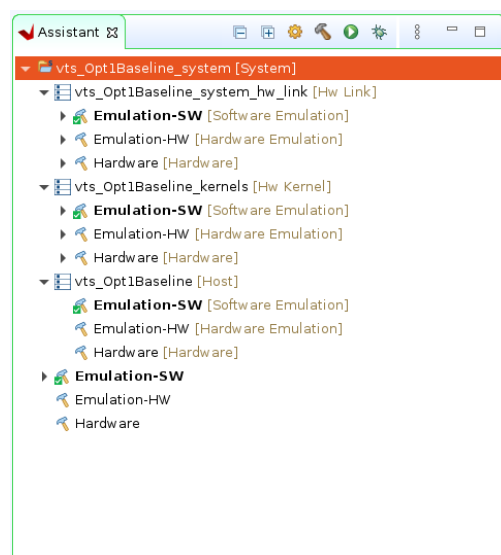
必須依照kernel → hw_link → host → system的順序來建置專案！

首先在Assistant中選擇kernel專案底下的Emulation-SW，接著按下上方Build project按鈕進行建置。



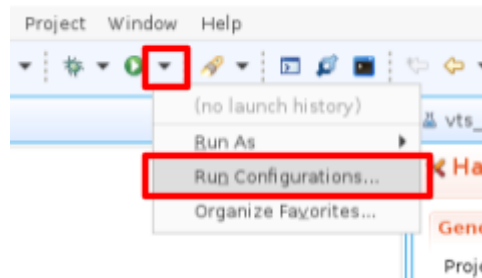
kernel建置完成後再依前述順序點選其他專案的Emulation-SW並進行建置，system的部分請直接點選vts_Opt1Baseline_system[System]來建置。

完成建置後Assistant View顯示如下，建置成功會有綠色打勾標示。

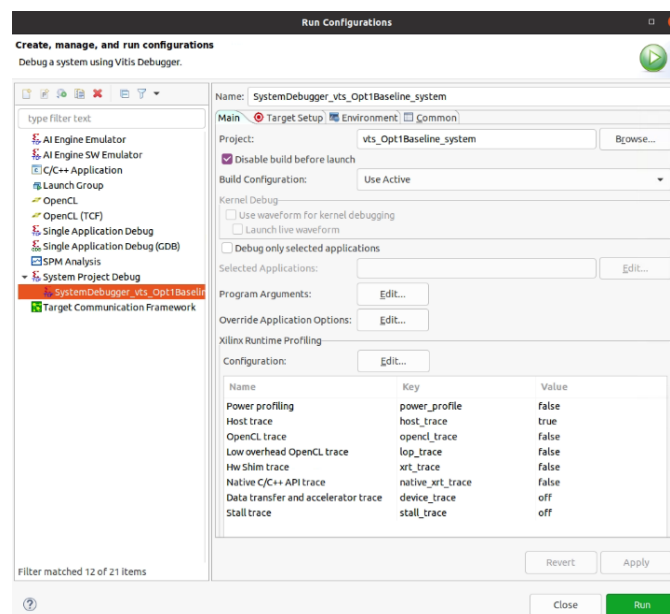
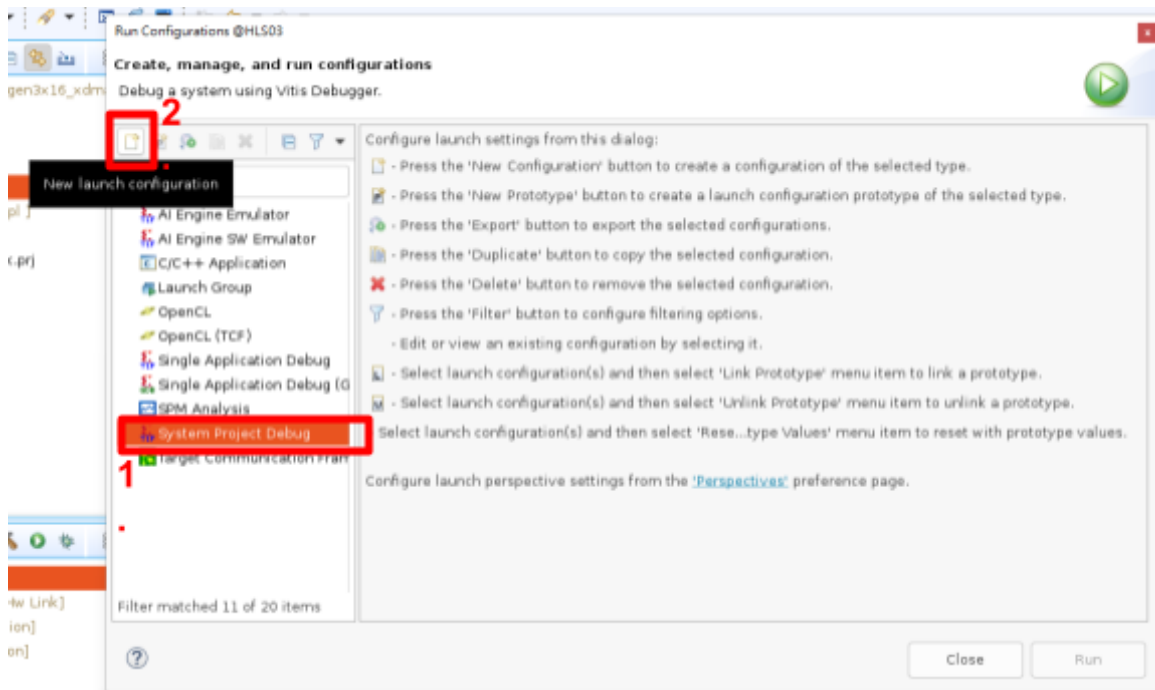


2.2.2. Run Emulation

執行emulation前要先設定其組態。點選Run Configurations

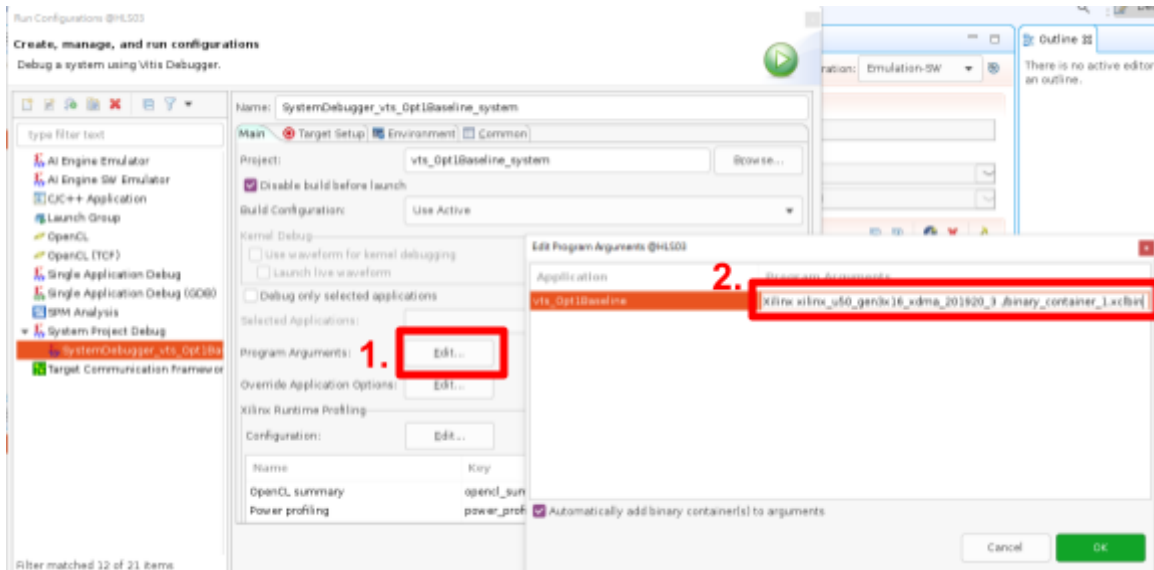


新增一個System Project Debug組態。

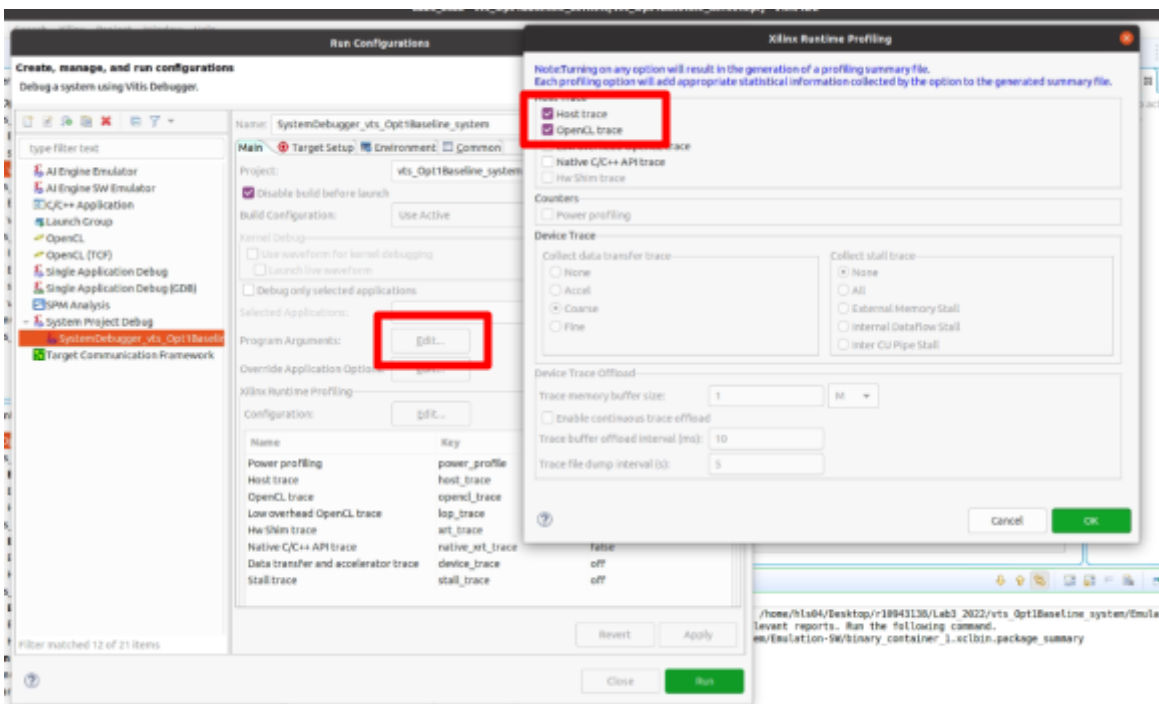


編輯Program Arguments, 提供host program所需的三個arguments:

Xilinx xilinx_u50_gen3x16_xdma_base_5 ./binary_container_1.xclbin
(用空白隔開這三項)

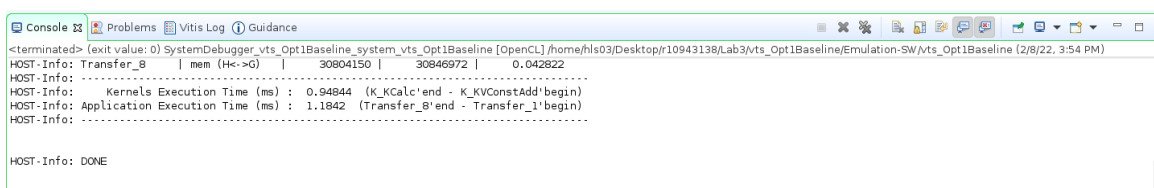


編輯Xilinx Runtime Profiling的Augments, 勾選Host trace 跟 OpenCL trace。



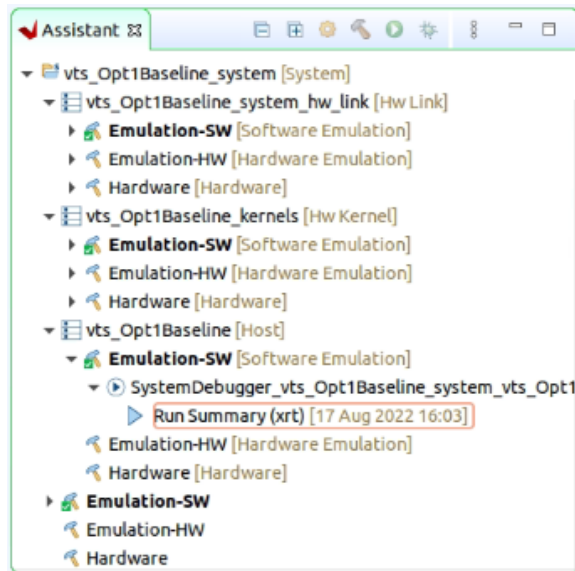
點選OK, 再點選Run開始執行software emulation。

Emulation完成後, 會在Console看到DONE訊息。

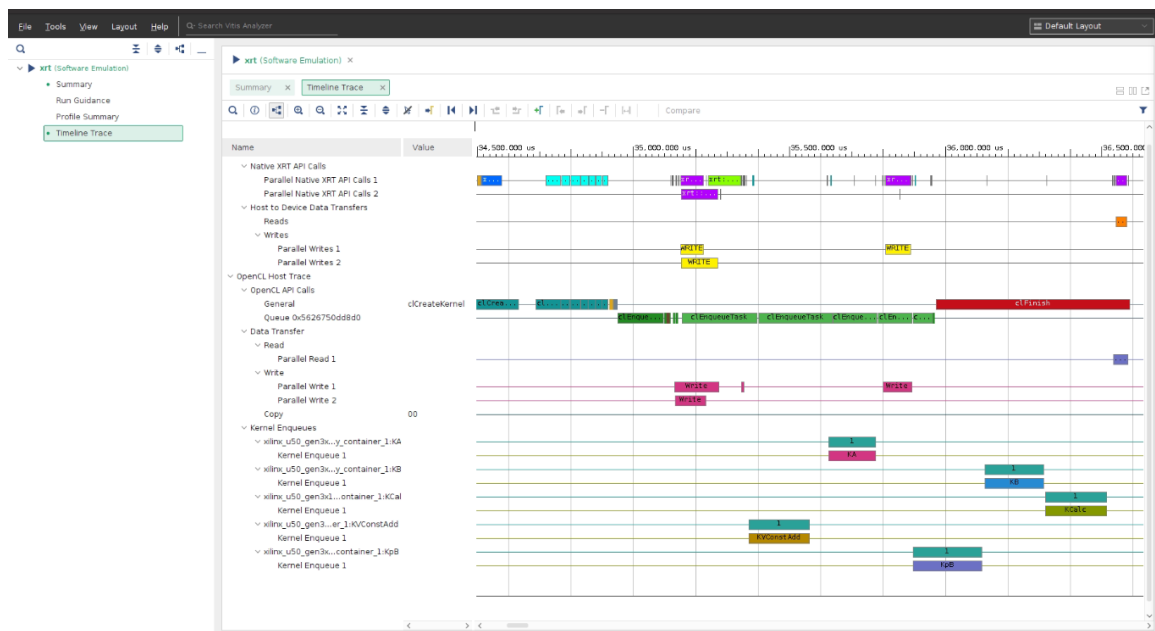


2.2.3. Analysis

Emulation執行完畢後，在Assistant View的vts_Opt1Baseline專案底下的Emulation-SW中會產生一個Run Summary，可以雙擊打開Vitis Analyzer查看各項report進行分析。



在Vitis Analyzer中，點選Application Timeline可查看host program以及kernel運行的時序。



2.3. Hardware Emulation

Hardware emulation是以軟體模擬XRT runtime到kernel FPGA的行為，類似於Lab. #1中的Co-simulation。

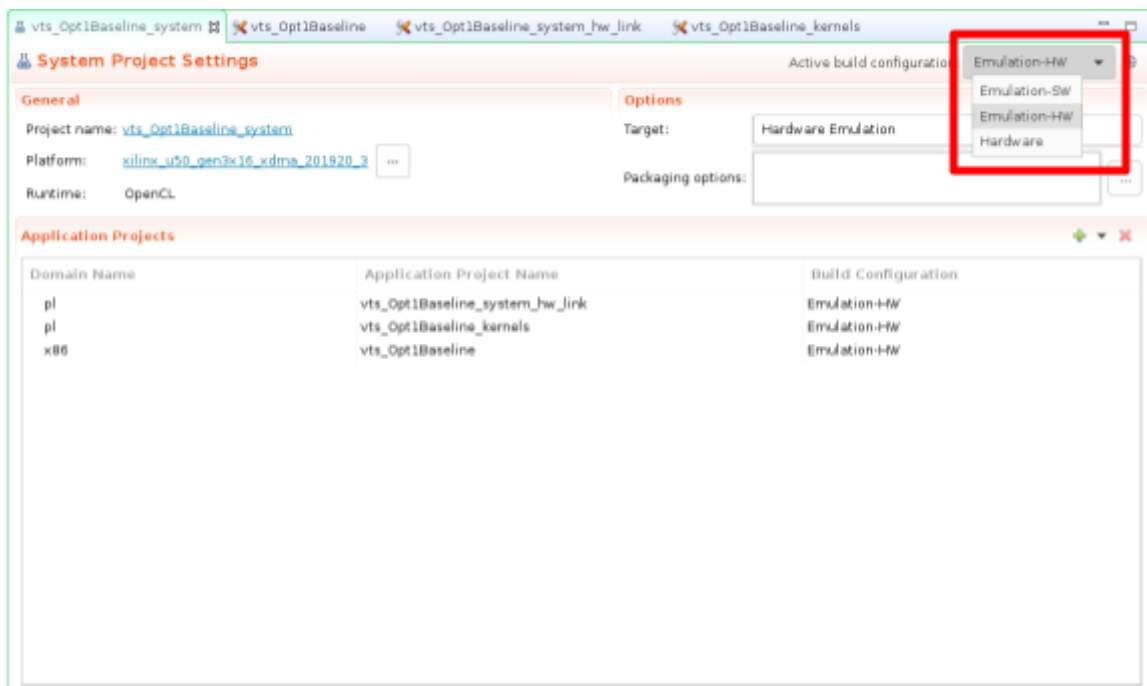
2.3.1. Build Project

步驟同Software Emulation，請參考前述步驟，改為點選Emulation-HW。過程視電腦配備可能需要數十分鐘。

(kernel → hw_link → host → system)

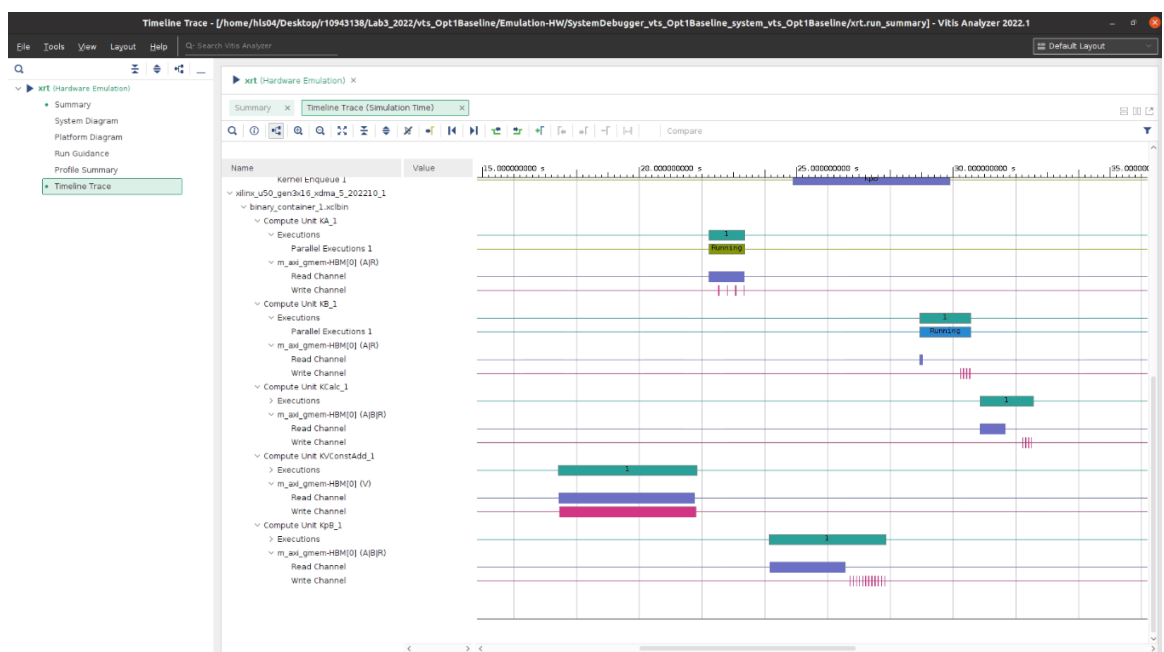
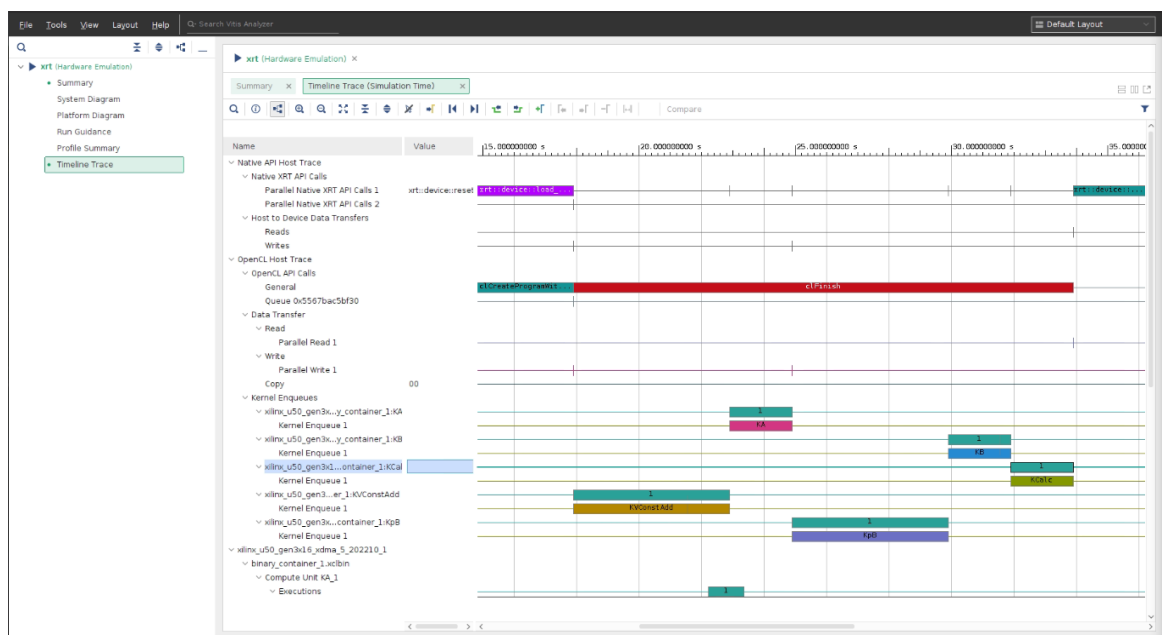
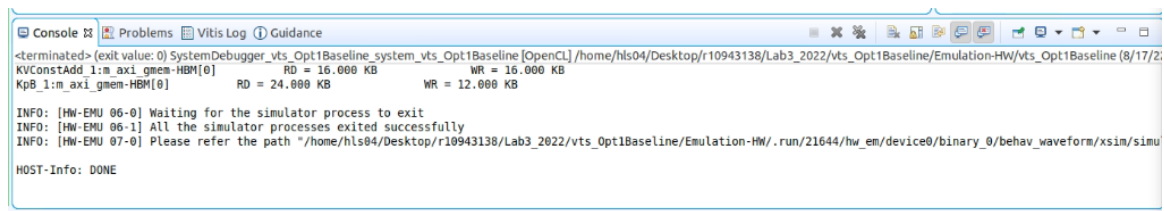
2.3.2. Run Emulation

請將Active build configuration設定為Emulation-HW，即可直接使用相同的Run Configuration毋須修改，其餘步驟與Software Emulation相同，請參考前述步驟。過程視電腦配備可能需要數十分鐘。



2.3.3. Analysis

步驟同Software Emulation, 請參考前述步驟, 打開在Emulation-HW中的Run Summary查看各項report進行分析。



2.4. Hardware

組態Hardware產生可在Alveo加速卡運行的程式 (host program) 及FPGA的 bitstream file (.xclbin)。

本實作Linux server上已建置Xilinx Alveo U50加速卡, 即可將Vitis建置好的host program及bitstream直接在伺服器中運行。

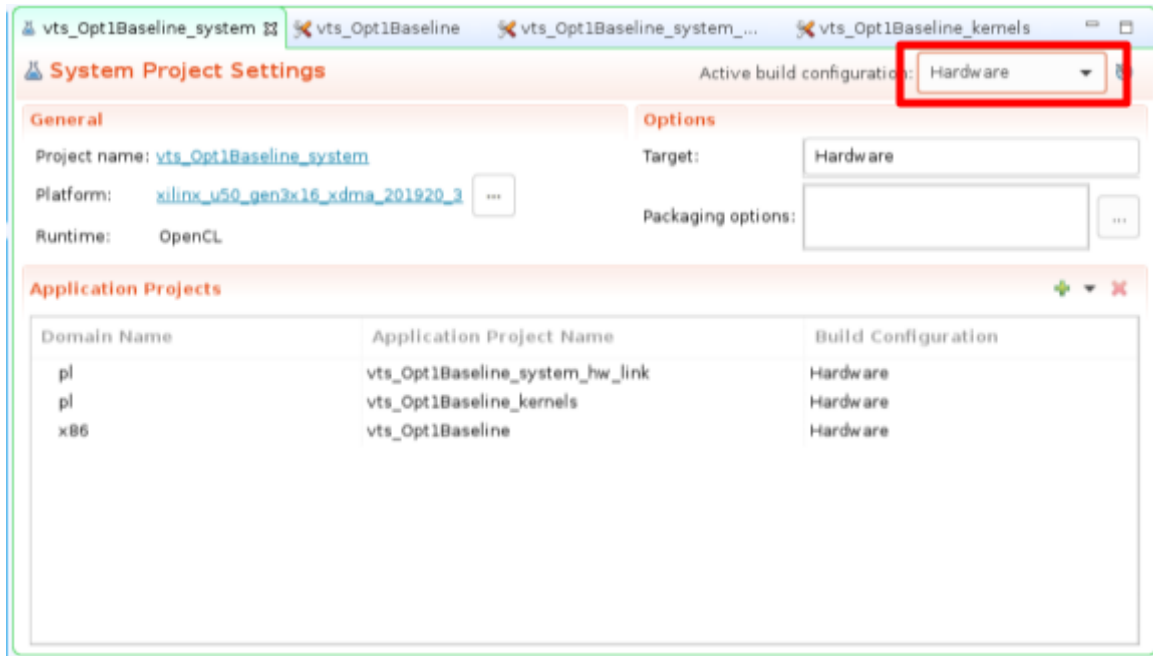
2.4.1. Build Project

步驟同Software Emulation, 請參考前述步驟, 改為點選Hardware。過程視電腦配備可能需要數十分鐘。(又比HW emulation更久)

(kernel → hw_link → host → system)

2.4.2. Run Hardware

請將Active build configuration設定為Hardware, 即可直接使用相同的Run Configuration毋須修改, 其餘步驟與Software Emulation相同, 請參考前述步驟。



2.4.3. Analysis

步驟同Software Emulation, 請參考前述步驟, 打開在Hardware中的Run Summary查看各項report進行分析。

