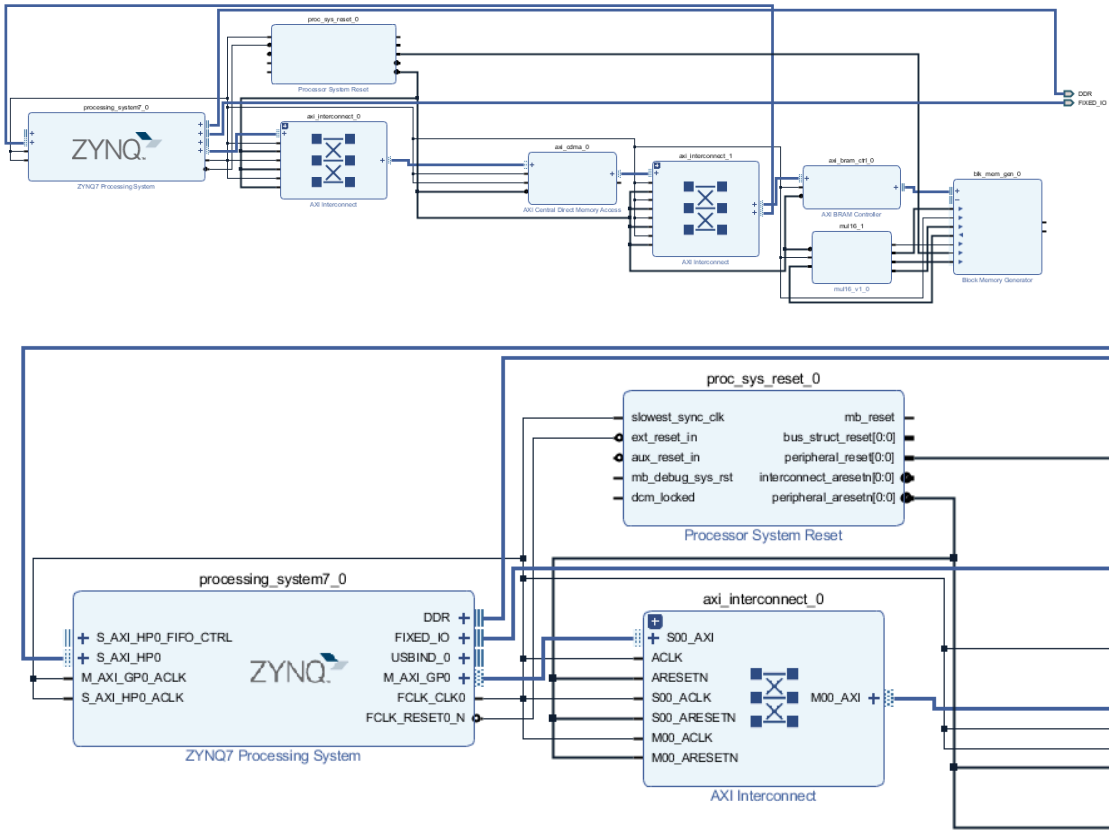
HOMEWORK 8

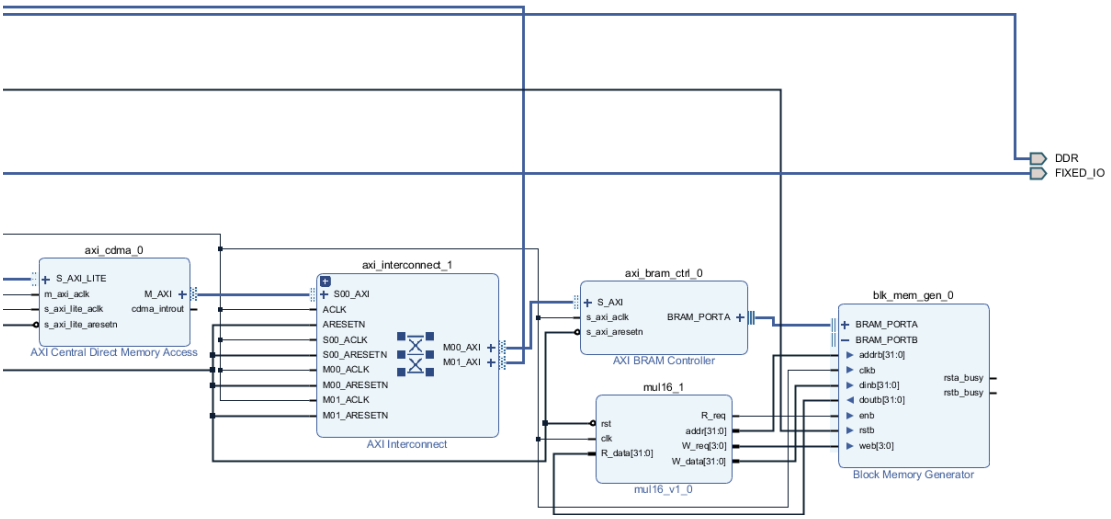
Student ID: P76134082、Q56134102　Name: 陳冠言 、王宇軒

Block design Screenshot:
(Please attach a screenshot and describe the block design function.)



1. processing_system7_0 (ZYNQ PS - 處理系統):
   - M_AXI_GP0 (Master General Purpose AXI Interface):ZYNQ作為主控端，透過此接口連接到axi_interconnect_0，用來設定和控制其他AXI週邊，例如CDMA控制器。
   - S_AXI_HP0 (Slave High Performance AXI Interface):ZYNQ作為從屬端，透過此接口連接到axi_interconnect_0。這使得FPGA中的AXI主控端(如CDMA)能夠高速存取ZYNQ的DDR記憶體。作業「注意事項1」中提到需要開啟此接口。
   - FCLK_CLK0:提供給FPGA邏輯部分的主要時脈信號。
   - DDR 和 FIXED_IO:ZYNQ與外部DDR記憶體和固定IO腳位的連接。
2. Processor System Reset:
   - 負責產生同步的重置信號給系統中的各個IP。
3. axi_interconnect_0 和 axi_interconnect_1 (AXI Interconnect):
   - AXI總線的交換中心，負責將多個AXI主控端連接到多個AXI從屬端。它們處理地址解碼、仲裁等。

- axi_interconnect_0: 主要連接ZYNQ的M_AXI_GP0 (主) 和 S_AXI_HP0 (從) 到CDMA的S_AXI_LITE (從) 和 M_AXI_... (主, 用於存取ZYNQ DDR)。
- axi_interconnect_1: 主要連接CDMA的M_AXI_... (主, 用於存取 BRAM) 到 axi_bram_ctrl_0 (從)。



1. axi_cdma_0 (AXI Central DMA):
   - DMA控制器, 負責在記憶體空間之間搬移資料。
   - S_AXI_LITE:CDMA的從屬控制接口, 連接到axi_interconnect_0。ZYNQ透過此接口設定CDMA的來源位址、目標位址、傳輸長度等參數。
   - M_AXI_... (例如 M_AXI_SG 或 M_AXI_MM2S/M_AXI_S2MM): CDMA的主控資料傳輸接口。當CDMA進行資料搬移時, 它會透過此接口讀取來源記憶體或寫入目標記憶體。這個主控接口會連接到:
     - axi_interconnect_0, 進而存取ZYNQ的S_AXI_HP0接口 (即DDR記憶體)。
     - axi_interconnect_1, 進而存取Block Memory (透過 axi_bram_ctrl_0)。
   - cdma_introut: 中斷信號, 通知ZYNQ資料傳輸完成或發生錯誤。
2. blk_mem_gen_0 (Block Memory Generator):
   - 用來生成FPGA內部的BRAM (Block RAM)。
3. axi_bram_ctrl_0 (AXI BRAM Controller):
   - 作為AXI總線和Block Memory原生接口之間的中介。
4. mul16_v1_0 (自訂的16位元乘法器IP):
   - 它不使用AXI接口, 而是直接連接到blk_mem_gen_0的 BRAM_PORTB。它會從Block Memory的特定位址(0x00和0x04)讀取資料, 相乘後將結果寫回Block Memory的特定位址(0x0C)。

Jupyter python code:

(Please describe the function and execution flow of the jupyter python code.)

```
from pynq import Overlay, MMIO
import warnings
warnings.filterwarnings("ignore")

axi_cdma_design = Overlay("./hw8.bit")
cdma_address = axi_cdma_design.ip_dict['axi_cdma_0']['phys_addr']

mem_addr = 0x30000000
bram_addr = 0xc0000000

in_addr = MMIO(mem_addr, 16)
cdma = MMIO(cdma_address, 44)

a=20
b=60
in_addr.write(0x0, a)
in_addr.write(0x4, b)

# CDMA move data to BRAM
cdma.write(0x00, 0x04)
cdma.write(0x18, mem_addr)
cdma.write(0x20, bram_addr)
cdma.write(0x28, 0x08)

cdma.write(0x00, 0x04)
cdma.write(0x18, bram_addr)
cdma.write(0x20, mem_addr)
cdma.write(0x28, 0x10)

print(f"ans of {a}*{b} = {in_addr.read(0xC)}") #1200
```

```
ans of 20*60 = 1200
```

1. 載入硬體設計(**bitstream**)

首先用 Overlay 將事先在 Vivado 中產生的 hw8.bit 硬體設計載入進 FPGA。這代表你燒錄了包含 ZYNQ + CDMA + BRAM + mul16 的設計。

2. 查詢與設定硬體元件地址

從設計中查詢 axi_cdma_0的記憶體起始位址，定義兩個重要區段的實體地址：

- 一塊給處理器使用的 DRAM 區(0x30000000)，用來暫存資料。
- 一塊 BRAM 區(0xC0000000)，是給乘法器存取的記憶體。

3. 初始化控制器

使用 MMIO 函式建立對指定記憶體區塊的控制權限，讓 Python 程式可以讀寫指定的位址。

4. 將兩個要相乘的數值寫入記憶體

將 20 與 60 這兩個整數分別寫入處理器記憶體的 0x30000000 與 0x30000004。

5. 使用 **CDMA** 把資料搬到 **BRAM**

啟動 CDMA，把這兩個整數資料從處理器端搬移到 BRAM(乘法器會讀取 BRAM 中的資料進行運算)。

6. 使用 **CDMA** 把乘法結果搬回處理器記憶體

等乘法器計算完畢後，再次啟動 CDMA，把乘積結果（存放於 BRAM 的特定位址）搬回處理器的記憶體。

7. 顯示乘法結果

從處理器記憶體中的 0x3000000C 讀出乘法結果，並印出 20 * 60 = 1200。

Lesson learn

(Please write down the experience of completing this assignment, what you learned, and the points of difficulty.)

透過本次作業學到許多關於ZYNQ系統設計、FPGA以及軟硬體協同運作知識。

以下是本次作業的經驗學習與可能遇到的困難點：

- AXI CDMA 的運作與應用：
  - 學習到如何使用賽靈思(Xilinx)的AXI中央直接記憶體存取(CDMA) IP核心，在記憶體映射的來源位址與目的地位址之間進行高頻寬的資料傳輸。
  - 理解CDMA的控制方式，例如透過寫入控制暫存器(CDMACR)來啟動CDMA、設定來源位址(SA)、目的地位址(DA)以及傳輸位元組數(BTT)來控制資料搬移的過程。
- Block Memory Generator (BMG) 的使用：
  - 學習使用Xilinx的Block Memory Generator IP來生成FPGA內部嵌入式的BRAM資源，並了解其讀寫訊號與配置方式，例如設定為True Dual Port RAM以允許多個元件同時存取。
  - 了解AXI BRAM Controller如何將Block Memory的介面轉換成AXI協定，使其能與AXI匯流排上的其他元件溝通。
- 自定義IP的封裝與整合：
  - 學習將Verilog HDL程式碼封裝成Vivado可用的自定義IP。
  - 將自定義IP整合到Vivado的Block Design中，並與其他IP（如CDMA、BRAM Controller）進行連接。
- Vivado 硬體系統建置:
  - 熟悉在Vivado中建立包含ZYNQ7處理系統、AXI Interconnect、AXI GPIO、CDMA、BRAM Controller以及自定義IP的硬體系統架構。
  - 理解AXI匯流排在連接各個IP核心時所扮演的角色，以及如何透過AXI Interconnect自動或手動連接主(Master)從(Slave)介面1。
  - 學習設定各IP的記憶體映射位址(Address Mapping)，並避免位址衝突或寫入系統保護區。
- PYNQ 平台的軟體控制:
  - 學習如何在PYNQ環境中使用Python（MMIO函式庫）來與FPGA硬體互動。

- 透過Python程式碼載入Overlay (bitstream), 並讀寫記憶體映射I/O (MMIO)位址來控制CDMA的運作及存取記憶體資料。
- 實現資料從ZYNQ系統記憶體搬移至BRAM, 由硬體（自定義IP）處理後, 再由CDMA將結果搬回ZYNQ系統記憶體的完整流程。

- CDMA 的控制邏輯:
  - 理解啟動CDMA並使其正確搬移資料的步驟順序。例如, 需要先寫入CDMACR來啟動CDMA, 再依序設定SA、DA, 最後寫入BTT來開始傳輸。
  - 設定正確的傳輸位元組數(BTT)。例如, 從ZYNQ搬移兩個數字到BRAM至少需要8 bytes, 而將包含結果的資料從BRAM搬回ZYNQ則至少需要16 bytes。