# 任務

**任務：**驗證這些32位元（0x0042a303）是否對應於RISC-V架構中的指令lw t1,4(t0)。

**0x0042a303 🡪 000000000100 00101 010 00110 0000011**

**imm11:0 = 000000000100**

**rs1 = 00101 = x5 (t0)**

**funct3 = 010**

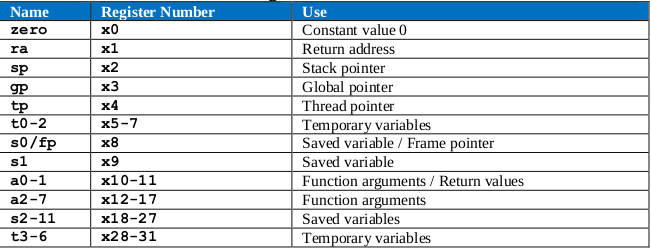
**rd = 00110 = x6 (t1)**

**op = 0000011**

來自DDCARV的附錄B：





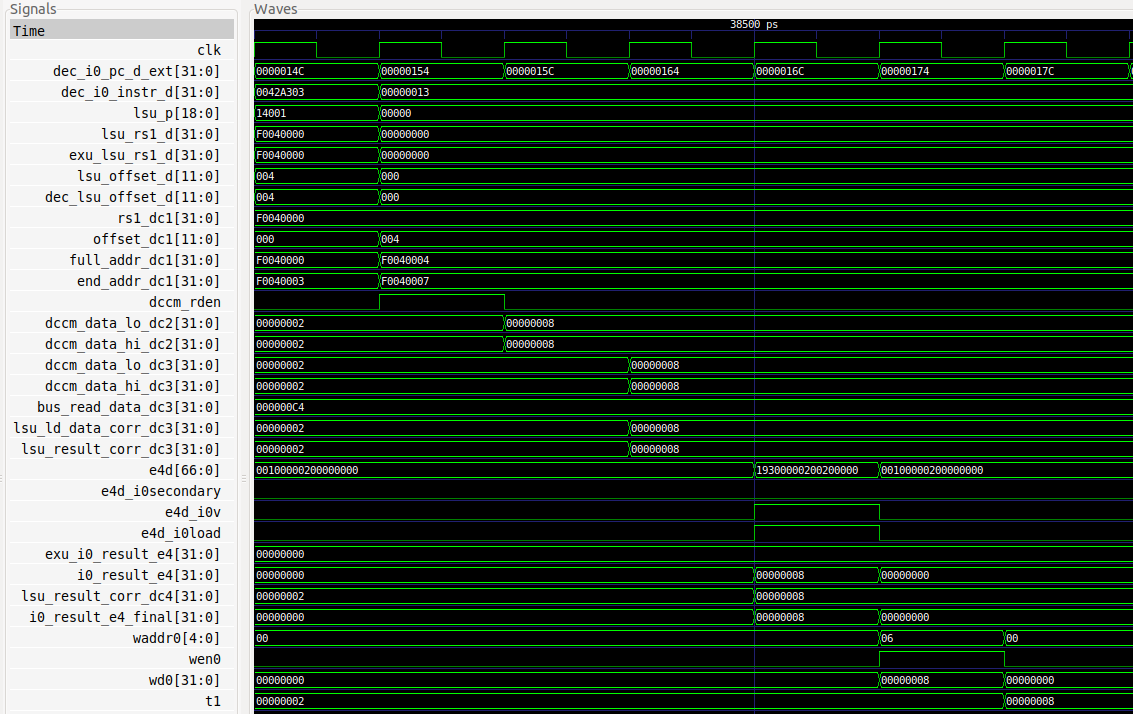


**任務：**在自己的電腦上重複圖4中的模擬過程。請按照以下步驟操作（如GSG第7部分所詳述）：

* 必要時產生模擬二進位檔案（*Vrvfpgasim*）。
* 在PlatformIO中，開啟在以下位置提供的專案：*[RVfpgaPath]/RVfpga/Labs/Lab13/LW\_Instruction\_DCCM*。
* 在檔案*platformio.ini*中更正到RVfpga模擬二進位檔案（*Vrvfpgasim*）的路徑。
* 使用Verilator產生模擬軌跡（產生軌跡）。
* 使用GTKWave開啟軌跡。
* 使用檔案*scriptLoad.tcl*（在*[RVfpgaPath]/RVfpga/Labs/Lab13/LW\_Instruction\_DCCM/*中提供）開啟與圖4所示訊號相同的訊號。為此，在GTKWave上，按一下「*File → Read Tcl Script File*」（檔案 → 讀取Tcl指令碼檔案）並選擇*scriptLoad.tcl*檔案。
* 按幾次「*Zoom In*」（放大）（）移動至18600 ps。

解答請參見實驗13的主文件。

**任務：**擴展圖4中的模擬以包含圖6所示的訊號（在下文說明）。



**任務：**在SweRV EH1處理器的Verilog檔案中找到圖6中的結構和訊號。

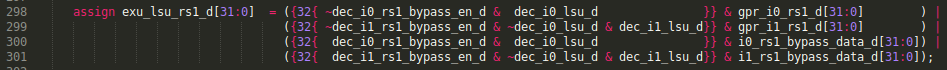
不提供解答。

**任務：**在圖4的模擬中包含訊號*lsu\_p*並根據上述說明分析其各個位元。

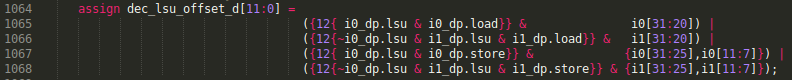
請參見上面的模擬。可以看到，當載入進行解碼時，lsu\_p = 0x14001：

* valid = 1。指令有效。
* load = 1。指令為載入指令。
* word = 1。存取的大小是字。

**任務：**在Verilog程式碼中從LSU的兩個輸入（exu\_lsu\_rs1\_d和dec\_lsu\_offset\_d）的取得來源分析這兩個輸入所遵循的路徑。此過程涉及幾個模組：**dec**、**exu**和**lsu**。為其他指令分析這些訊號的行為。



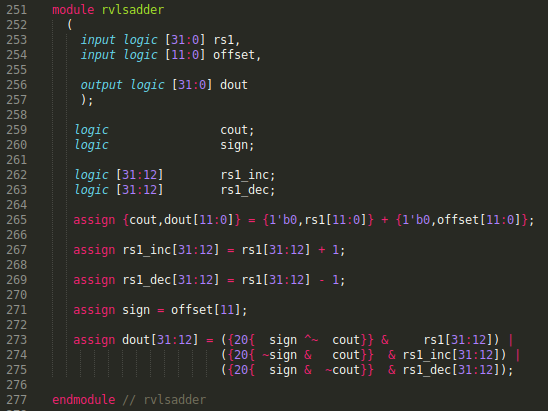
基本位址可以來自暫存器檔案或旁路（來自通路0或通路1）。



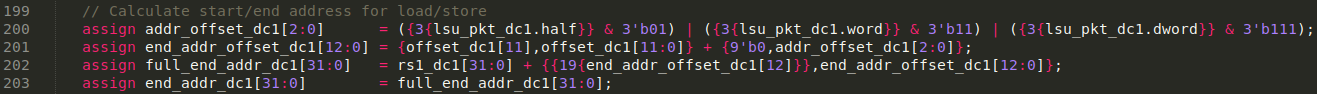
偏移量來自通路0或通路1指令的32位元。

**任務：**分析DC1階段中兩個加法器的實作，這兩個加法器在模組**lsu\_lsc\_ctl**中實例化。我們透過展示這些加法器的實作在下面的圖7中提供指導。

**檔案*beh\_lib.sv*：**

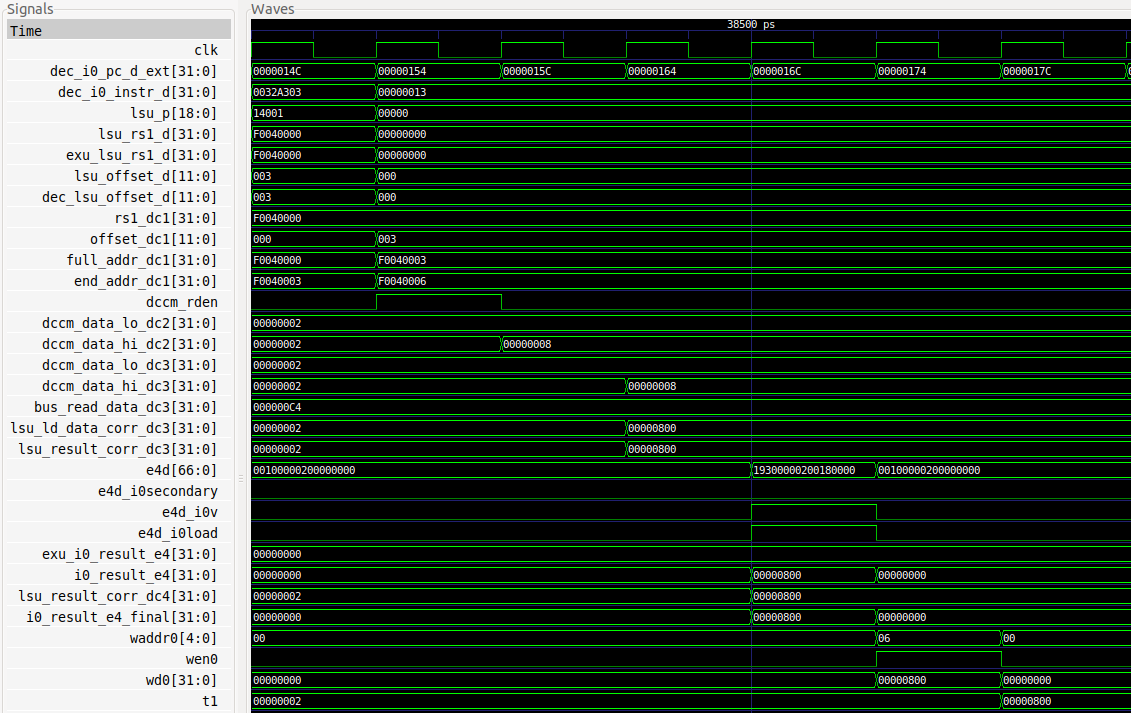


**檔案*lsu\_lsc\_ctl.sv*：**



**任務：**在圖2的程序中，嘗試不同的存取大小（位元組和半字）和未對齊存取。為此，請變更偏移量或將存取類型從lw變更為lb（*載入位元組*）或lh（*載入半字*）。例如，如果將偏移量從4變更為3，則載入字指令將對從位址0xF0040003開始的32位元執行未對齊存取，如圖8所示。分析上述不同情況下訊號lsu\_addr\_dc1[31:0]（或full\_addr\_dc1[31:0]）和end\_addr\_dc1[31:0]的值。

在實驗20中，我們將從DCCM的內部分析這種情況。



訊號lsu\_addr\_dc1[31:0]和end\_addr\_dc1[31:0]的值將存取的起始位址和結束位址傳達給記憶體：0xF0040003和0xF0040007。讀取兩個字（0x00000002和0x00000008），並在對齊器中擷取最後一個字（0x00000800）。

**任務：**在圖2的程序中，當對位址0xF0040004和位址0xF0040003執行lw時，比較訊號dccm\_data\_lo\_dc2[31:0]和dccm\_data\_hi\_dc2[31:0]的值。

上文有兩個模擬。

* 指向位址0xF0040004的lw

dccm\_data\_lo\_dc2[31:0]：0x00000008

dccm\_data\_hi\_dc2[31:0]：0x00000008

兩個訊號都包含從要求位址讀取的值。

* 指向位址0xF0040003的lw

dccm\_data\_lo\_dc2[31:0]：0x00000002（來自位址0xF0040000的值）

dccm\_data\_hi\_dc2[31:0]：0x00000008（來自位址0xF0040004的值）

**任務：**分析**lsu\_dccm\_ctl**和**lsu\_ecc**模組中的Verilog程式碼中使用的對齊、合併和錯誤檢查邏輯。

不提供解答。

**任務：**在圖2的程序中，當對位址0xF0040004和位址0xF0040003執行lw時，比較訊號lsu\_result\_corr\_dc3[31:0]的值。

上文有兩個模擬。

* 指向位址0xF0040004的lw

lsu\_result\_corr\_dc3[31:0]：0x00000008

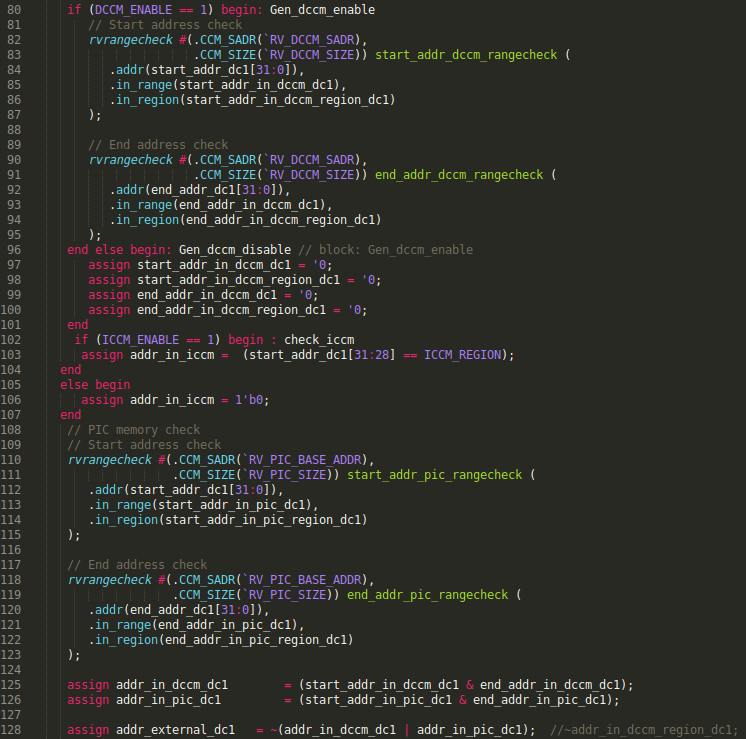
它包含從要求位址讀取的值。

* 指向位址0xF0040003的lw

lsu\_result\_corr\_dc3[31:0]：0x00000800

它包含從要求位址讀取的值。需考慮RISC-V採用位元組由小到大模式。

**任務：**在Verilog程式碼中分析訊號addr\_external\_dc1如何於DC1階段在模組**lsu\_addrcheck**中計算。



模組**rvrangecheck**用於檢查要求位址：

* 如果它處於DCCM/ICCM位址範圍中（第80-107行），在這種情況下，訊號addr\_in\_dccm\_dc1 = 1
* 如果它處於PIC位址範圍中（第108-123行），在這種情況下，訊號addr\_in\_pic\_dc1 = 1
* 如果它不處於上述任一位址範圍中，則處於DDR外部記憶體中，在這種情況下：addr\_external\_dc1 = 1

**任務：**驗證這些32位元（0x0062a023）是否對應於RISC-V架構中的指令sw t1,0(t0)。

**0x0062a023 🡪 0000000 00110 00101 010 00000 0100011**

**imm11:0 = 000000000000**

**rs2 = 00110 = x6 (t1)**

**rs1 = 00101 = x5 (t0)**

**funct3 = 010**

**op = 0100011**

來自DDCARV的附錄B：

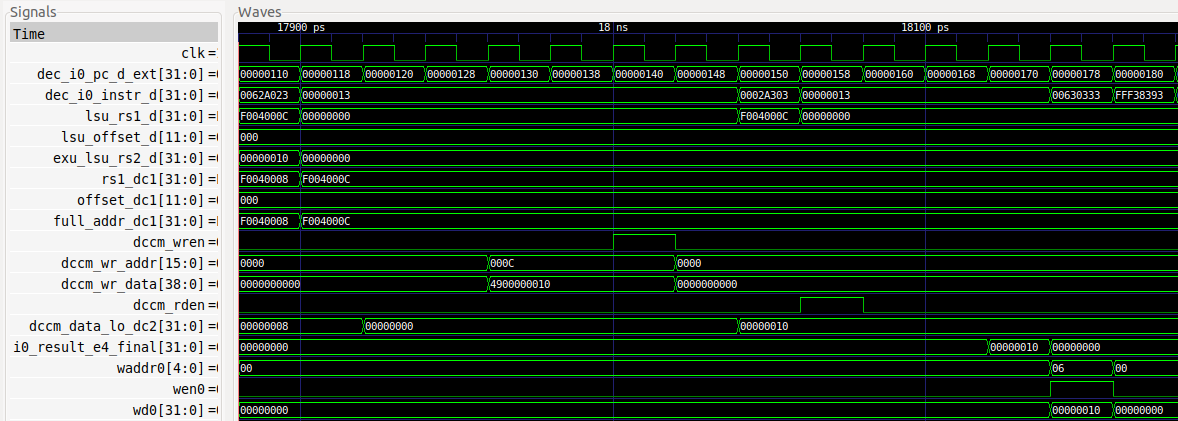


**任務：**在自己的電腦上重複圖12中的模擬過程。請按照以下步驟操作（如GSG第7部分所詳述）：

* 必要時產生模擬二進位檔案（*Vrvfpgasim*）。
* 在PlatformIO中開啟在以下位置提供的專案：*[RVfpgaPath]/RVfpga/Labs/Lab13/SW\_Instruction\_DCCM*。
* 在檔案*platformio.ini*中更新到RVfpga模擬二進位檔案（*Vrvfpgasim*）的路徑。
* 使用Verilator產生模擬軌跡（產生軌跡）。
* 在GTKWave上開啟軌跡。
* 使用檔案*scriptStore.tcl*（在*[RVfpgaPath]/RVfpga/Labs/Lab13/SW\_Instruction\_DCCM/*中提供）顯示與圖4所示訊號相同的訊號。為此，在GTKWave上，按一下「*File → Read Tcl Script File*」（檔案 → 讀取Tcl指令碼檔案）並選擇*scriptStore.tcl*檔案。
* 按幾次「*Zoom In*」（放大）（）移動至17900 ps。

解答請參見實驗13的主文件。

**任務：**在模擬中分析儲存指令之後的載入指令，以驗證值是否已正確寫入DCCM。需要新增圖4和圖6中的一些訊號來分析載入。



**任務：**按照與第2.B部分中對lw指令執行的進階分析類似的方式擴展本部分中對sw指令執行的基礎分析。

不提供解答。

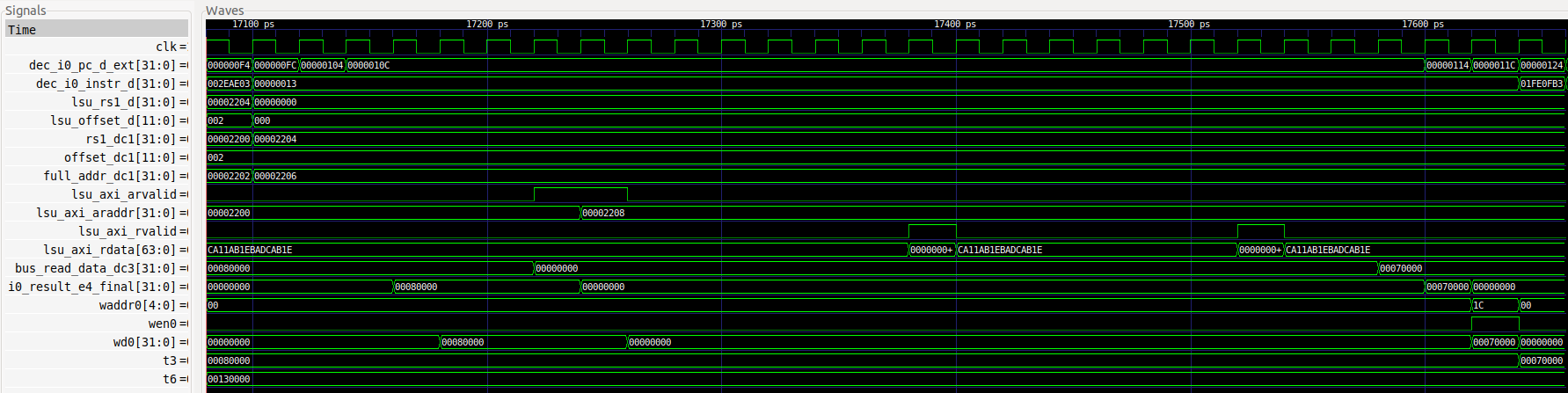
**任務：**分析針對DCCM的未對齊儲存以及子字儲存：儲存位元組（sb）或*儲存半字*（sh）。

不提供解答。

**任務：**在自己的電腦上重複圖17中的模擬過程。使用檔案*test\_Blocking.tcl*（在*[RVfpgaPath]/RVfpga/Labs/Lab13/LW\_Instruction\_ExtMemory*中提供）。按幾次「*Zoom In*」（放大）（）移動至16940 ps。

解答請參見實驗13的主文件。

**任務：**修改圖15中的程序以分析需要透過AXI匯流排向外部記憶體傳送兩個位址的未對齊載入存取。

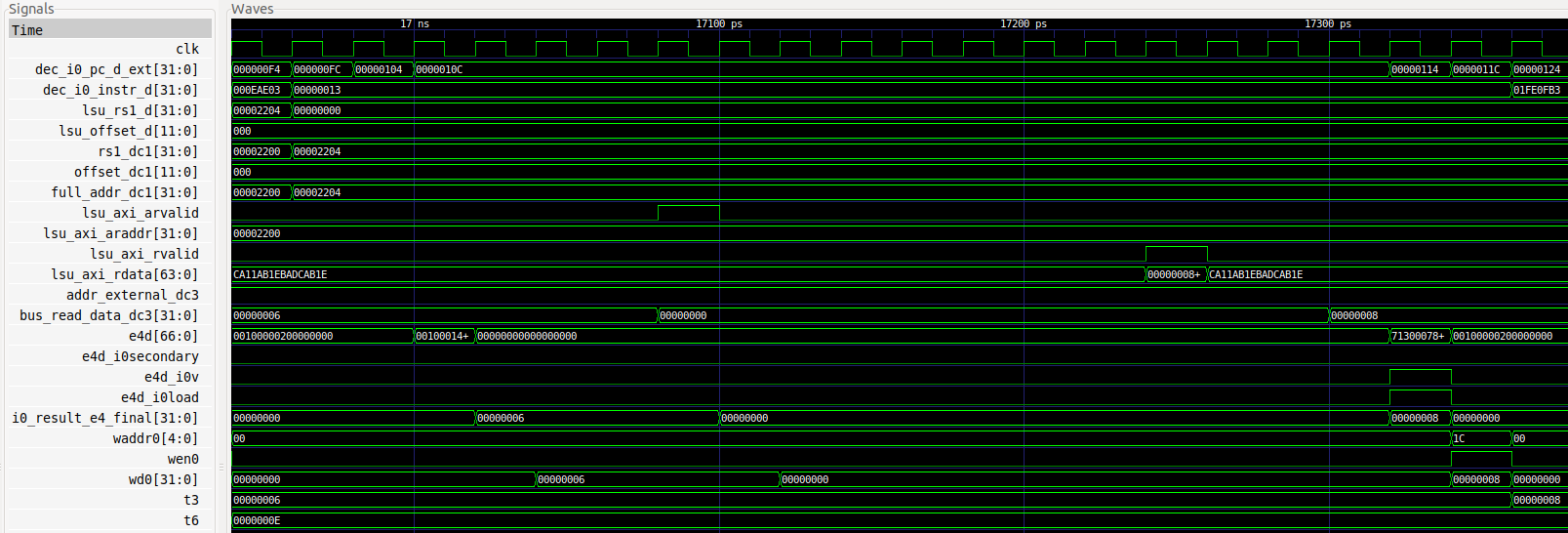


**任務**：將控制多路開關的訊號新增到模擬中（在圖16的DC3和提交階段），其中多路開關選擇由DDR外部記憶體提供的資料。可以在Verilog程式碼的以下幾行中找到這些多路開關：

- 2:1多路開關：模組**lsu\_lsc\_ctl**的第264行。

- 3:1多路開關：模組**dec\_decode\_ctl**的第2277行。

可以使用的*.tcl*檔案位於：*[RVfpgaPath]/RVfpga/Labs/Lab13/LW\_Instruction\_ExtMemory/test\_Blocking\_Extended.tcl*



**任務：**分析用於存取DRAM控制器的AXI匯流排實作也很有趣，為此可以檢查**lsu\_bus\_intf**模組。

不提供解答。

**任務：**在自己的電腦上重複圖18中的模擬過程。使用檔案*scriptStoreBuffer.tcl*（在*[RVfpgaPath]/RVfpga/Labs/Lab13/SW\_Instruction\_DCCM*中提供）。按幾次「*Zoom In*」（放大）（）移動至17900 ps。

解答請參見實驗13的主文件。

**任務：**修改圖11中的程序以實作兩個出色的儲存操作，並執行與圖18中的分析類似的分析。

不提供解答。