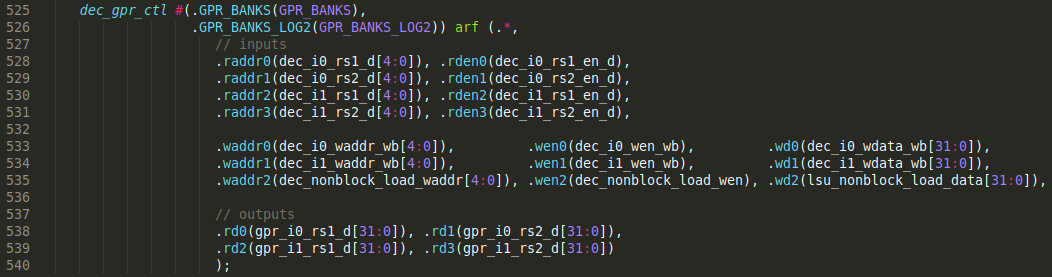
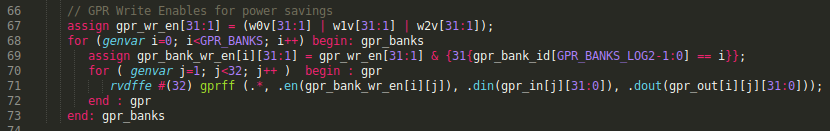
# 1. 任務

**任務：**在模組**dec\_gpr\_ctl**中實作暫存器檔案，並在模組**dec**中將其實例化（參見圖7）。  
分析模組**dec\_gpr\_ctl**的Verilog程式碼及主要訊號的模擬（位於檔案*[RVfpgaPath]/RVfpga/src/SweRVolfSoC/SweRVEh1CoreComplex/dec/dec\_gpr\_ctl.sv*中），以瞭解其工作方式。請注意，SweRV EH1處理器允許包含多個暫存器檔案，但RVfpga系統中使用的組態僅使用一個暫存器檔案（參見檔案*dec.sv*的第402行：  
localparam GPR\_BANKS = 1;）。

**模組dec中的實例化**：

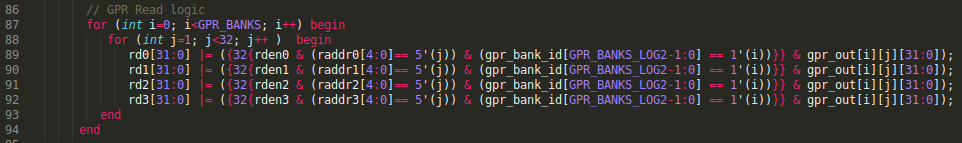


**模組dec\_gpr\_ctl中32個暫存器的實作**：



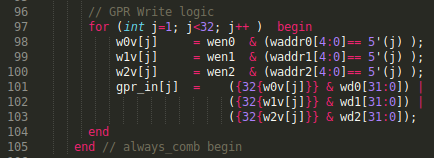
本例中僅實作了1個記憶庫。這一記憶庫透過將模組**rvdffe**（位於檔案*[RVfpgaPath]/RVfpga/src/SweRVolfSoC/SweRVEh1CoreComplex/lib/beh\_lib.sv*中）實例化31次實作31個暫存器。注意每個**rvdffe**暫存器的寬度使用參數選擇，本例中為32位元🡪 rvdffe #(32)。暫存器0不是必需的，因為RISC-V架構強制其始終為0。

**暫存器讀取操作**：



實作了4個讀取連接埠。每一個讀取連接埠均分配有暫存器的值（用raddr0/raddr1/raddr2/raddr3訊號表示）。rden0/rden1/rden2/rden3訊號啟用/停用讀取操作。注意*j*的初始值為1，因此暫存器0的讀取操作始終傳回值0。

**暫存器寫入操作**：



實作了3個寫入連接埠。每個暫存器寫入訊號wd0/wd1/wd2中提供的值，具體取決於暫存器位址waddr0/waddr1/waddr2。wen0/wen1/wen2訊號啟用/停用寫入操作。注意*j*的初始值為1，因此沒有寫入暫存器0。

**任務：**從圖8分析多工器的控制位元。請注意，控制位元在訊號e3d中，該訊號由訊號dd經管線處理得到，後一個訊號由控制單元在解碼階段產生（有關控制位元的說明，請參見SweRVref.docx）。

* 如果DC3階段的指令有效（e3d.i0v == 1）且為load指令（e3d.i0load == 1），則選擇來自LSU管線的值：i0\_result\_e3\_final = lsu\_result\_dc3。
* 如果EX3階段的指令有效（e3d.i0v == 1）且為mul指令（e3d.i0mul == 1），則選擇來自乘法器的值：i0\_result\_e3\_final = exu\_mul\_result\_e3。
* 否則，將選擇來自I0管線的值：i0\_result\_e3\_final = i0\_result\_e3。

**任務：**從圖9分析多工器的控制位元，這些控制位元位於模組**dec\_decode\_ctl**中。

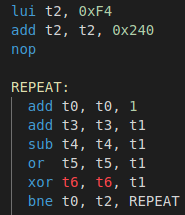
* 如果EX4階段的結果必須從I0輔助ALU中選擇（e4d.i0secondary == 1），則選擇來自I0輔助ALU的值：i0\_result\_e4\_final = exu\_i0\_result\_e4。我們將在實驗15中分析輔助ALU操作。
* 如果DC4階段的指令有效（e4d.i0v == 1）且為load指令（e4d.i0load == 1），則選擇來自LSU管線的值：i0\_result\_e4\_final = lsu\_result\_corr\_dc4。
* 否則，將選擇來自I0管線的值：i0\_result\_e4\_final = i0\_result\_e4。

**任務：**按照以下步驟（如GSG第7部分中詳述）在自己的電腦上重複圖11和圖12中的模擬  
過程：

* 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/Lab11/ExampleProgram*。
* 在檔案*platformio.ini*中建立到RVfpga模擬二進位檔案（*Vrvfpgasim*）的正確路徑。
* 使用Verilator產生模擬軌跡（產生軌跡）。
* 使用GTKWave開啟軌跡。
* 使用檔案*test\_1.tcl*和*test\_2.tcl*（在*[RVfpgaPath]/RVfpga/Labs/Lab11/ExampleProgram*中提供）開啟與圖11和圖12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），然後選擇*test\_1.tcl*或*test\_2.tcl*檔案。
* 按幾次「*Zoom In*」（放大）（）移動至48500 ps（或迴圈的任何其他迭代，第一次迭代除外）。

解答請參見實驗11的主文件。

**任務：**按照GSG所述在Nexys A7板上執行圖13中的程式。對於測量的四個事件，應獲得圖14所示的結果。解釋並證明結果。



程式由1000000次迭代的迴圈構成，該迴圈包含5條算術邏輯指令和一個條件分支。未發生由於冒險引起的暫停，因此：

* + 執行6 \* 1000000條指令
  + 每個週期執行2條指令，因此：(6/2) \* 1000000個週期
  + 執行1000000個分支，據預測幾乎所有分支均命中。

**任務：**在圖13所示程式的硬體計數器中測量其他事件。為此，必須使用pspPerformanceCounterSet函數在*Test.c*檔案中變更待測量事件的組態。請注意，可以使用WD PSP檔案中定義的巨集參考不同的事件（如圖1所示）：  
*.platformio/packages/framework-wd-riscv-sdk/psp/api\_inc/psp\_performance\_monitor\_eh1.h*。例如，如果要測量I$未命中數而不是分支未命中數，則必須在檔案中將*Test.c*行：**pspPerformanceCounterSet**(D\_PSP\_COUNTER3, E\_BRANCHES\_MISPREDICTED);

替換為行：**pspPerformanceCounterSet**(D\_PSP\_COUNTER3, E\_I\_CACHE\_MISSES);

不提供解答。

**任務：**在Test\_Assembly函數中提供其他程式並檢查不同的事件是否提供了預期的結果。可以嘗試其他指令，例如載入、儲存、乘法、除法...以及引發管線暫停的冒險。

不提供解答。