VLSI System Design (Graduate Level)

Fall 2024

HOMEWORK III

REPORT

Must do self-checking before submission:

Compress all files described in the problem into one tar

All SystemVerilog files can be compiled under SoC Lab environment

All port declarations comply with I/O port specifications

Organize files according to File Hierarchy Requirement

No any waveform files in deliverables

Student name: \_王劭懷\_\_\_\_\_\_\_\_ , \_\_\_\_\_\_\_\_\_

Student ID: \_M16121077\_\_\_\_\_\_, \_\_\_\_\_\_\_

**Summary**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Hardware | | | | | | | | |
|  | | | | | | RTL | | synthesis |
| Top | CPU\_wrapper | | CPU | | | PASS | | PASS |
| New instructions | | | PASS | | PASS |
| SRAM\_wrapper (IM & DM) | | | | | PASS | | PASS |
| ROM\_wrapper | | | | | PASS | | PASS |
| DRAM\_wrapper | | | | | PASS | | PASS |
| AXI | | | | | PASS | | PASS |
| DMA | | | | | PASS | | PASS |
| Watch Dog Timer | | | | | PASS | | PASS |
| Synthesis result | | | | | | | | |
| Area | | | | | Clock cycle(ns) | | | |
| 21029.226425 | | | | | 1 | | | |
| Firmware & Software | | | | | | | | |
|  | | RTL pass | | syn pass | | | Execution time(ns) | |
| Booting | | PASS | | PASS | | | - | |
| Prog 0 | | PASS | | PASS | | | 38506 | |
| Prog 1 | | PASS | | PASS | | | 1370182 | |
| Prog 2 | | PASS | | PASS | | | 1421378 | |
| Prog 3 | | PASS | | PASS | | | 907340 | |
| Prog 4 | | PASS | | PASS | | | 908329 | |
| Prog 5 | | PASS | | PASS | | | 29610 | |
| Spyglass summary(number of inline messages) | | | | | | | | |
| Information | | Warning | | Error | | | Fatal | |
| 148 | | 3 | | 0 | | | 0 | |
| Superlint(number of inline messages) | | | | | | | | |
| Total lines | | Warning | | Error | | | coverage(%) | |
| 7276 | | 52 | | 1 | | | 99.27% | |

**Contribution**

|  |  |
| --- | --- |
| 王劭懷 100% | XXX 50% |
| ALL Design |  |

**Hardware Design Description**

* System Block Diagram

一張含有 文字, 圖表, 方案, 設計 的圖片

自動產生的描述

* Interrupt mechanism description and flow chart

在現有的 CPU 中新增了一個 CSR 模組，以支持額外的 CSR 指令操作，並處理中斷的相關機制。其中，DMA 和 WDT 的中斷處理方式分別設計如下： DMA 中斷處理 當 CPU 接收到 WFI 指令時，CSR 模組會辨識該指令，並讓 CPU 的所有 pipeline stage 進入 stall 狀態，靜待 DMA 中斷信號的到來。 收到 DMA 的中斷信號後，系統會檢查 CSR 的 mie 寄存器中對應的中斷允許位 (bit) 是否為 1。 如果允許位為 1，則將該中斷信號存入暫存器，並輸出 interrupt\_dma\_delay 信號供其他暫存器使用，用以準備新地址的資料讀取。 同時，程序計數器 (PC) 會被設置為中斷處理程序的 trap 地址。 由於在進入中斷前，上一個讀取的地址可能無效，導致不必要的指令被讀取。因此，在收到 DMA 中斷信號時，會清空 ID 階段的指令 (flush ID stage)，以避免執行非預期的操作。 進入中斷處理程序後，CPU 將執行 trap 中的相關指令，直到接收到 mret 指令為止。此時，PC 會回復到之前存儲於 mepc 寄存器中的地址，並再次 flush ID 階段的指令，以確保流程的正確性。 WDT 中斷處理 WDT 中斷的處理不一定是在 CPU 處於 WFI 狀態時觸發，但仍然通過 CSR 模組完成管理，其操作流程與 DMA 中斷類似。 不同之處在於： WDT 中斷會根據 CSR 模組中另一個寄存器位 (bit) 進行判斷。 返回地址的保存與恢復方式亦有些微差異。 這套設計有效整合了 CSR 模組對 WFI 指令、中斷處理及流程控制的管理，使 CPU 能夠根據不同情境動態調整運行邏輯，提高了系統的靈活性與可靠性。

一張含有 圖表, 方案, 工程製圖, 圖解 的圖片

自動產生的描述

WFI 時，CSR output WFI的signal給CPU其他暫存器，使其stall住，等待interrupt的訊號。

一張含有 螢幕擷取畫面, 軟體, 多媒體軟體, 鮮豔 的圖片

自動產生的描述

收到DMA interrupt 時進入trap，將pc改為trap\_pc(mtvec)，從主程式改為執行中斷程式。並將多讀取的指令從ID stage flush掉。再者，透過interrupt\_dma\_delay將CPU stall住等待trap\_pc的起始位址從記憶體索取data。

一張含有 螢幕擷取畫面, 行, 文字 的圖片

自動產生的描述

DMA interrupt後讀取mret的instruction時，PC改為原先存在mepc的PC值接續執行main program，並會在後面將DMAEN關閉。

一張含有 螢幕擷取畫面, 軟體, 多媒體軟體, 繪圖軟體 的圖片

自動產生的描述

WDT interrupt，在將WDT的WDEN寫為1前，會先將WTOCNT寫入欲計算的邊界值，而後做CDC的處理傳送到WDT內，WDT收到WDEN後開始執行計數器。

一張含有 螢幕擷取畫面, 電子產品, 多媒體軟體, 軟體 的圖片

自動產生的描述

WDT 與DMA的interrupt相同，會進入trap的地址開始執行。

一張含有 螢幕擷取畫面, 文字, 行, 鮮豔 的圖片

自動產生的描述

**一張含有 螢幕擷取畫面, 文字, 軟體, 多媒體軟體 的圖片

自動產生的描述**

WDLIVE在執行完第一次WTO後，會一直將其拉起使得WTD的counter歸零，讓其不會再傳送WTO使CPU進到interrupt的處理。

**一張含有 螢幕擷取畫面, 鮮豔, 電子產品, 電路 的圖片

自動產生的描述**

* DRAM wrapper FSM chart

Write

一張含有 行, 文字, 圖表, 圓形 的圖片

自動產生的描述

Read

一張含有 行, 文字, 圖表, 繪圖 的圖片

自動產生的描述

* WDT & CDC circuit description and diagram

**一張含有 圖表, 方案, 工程製圖, 行 的圖片

自動產生的描述**

我的CDC電路是透過單純的2D Flip-flop做處理，以解決亞穩態的問題，在欲寫入的訊號同時新增控制訊號，針對控制訊號做CDC的處理，等到控制訊號確定傳到clk2 domain時，才將data存入WDT的暫存器中做處理。而在快到慢的處理時，為了避免訊號消失，我透過counter來將原先的訊號 hold住超過clk2的一個完整的cycle，使得訊號必能傳到clk2的domain。

**Software & Firmware design description**

* Prog 1

定義timer\_interrupt\_handle、external\_interrupt\_handler，並定義trap\_handler用於區分中斷類型並調用對應的處理函數。主程式則負責複製資料陣列到目標記憶體區，並使用氣泡排序法對資料進行排序。而演算法的部分是外層的迴圈確保每次排序一個最大的元素到正確位址，內層迴圈則逐一比較並交換相鄰元素，確保順序正確，持續迴圈直到整個陣列排序完成。而如果發生計時器或外部中斷，trap handler會呼叫對應的處理器進行處理

* Prog 2

中斷處理與prog1相同，差異在主程式演算法的處理，先是定義了目標記憶體的起始地址、BMP圖像數據的起始和結束地址與BMP數據的大小。BMP的前54位是文件頭，直接複製到目標記憶體，接著對BMP圖像進行處理，從第55位開始對每個像素的RGB值進行以下處理，如果RGB值相同，將這三個值設定為相同的灰階值，如果RGB不同，計算加權平均數來生成灰階值，結果覆蓋原像素的RGB值，達到灰階化的效果。

* Booting

將DRAM中的指令搬移到處理器可直接存取的記憶體區域，包括指令記憶體、靜態資料區、靜態初始化資料區，並使用DMA減少處理器的負載並提高效能。將位於DRAM中的程式指令搬移到內部指令記憶體，計算搬移的數據大小，使用DMA控制器來源地址、目標地址和數據長度，然後啟用DMA，並在每次DMA操作之前，通過寫入mie register來啟用中斷，並啟用wfi指令讓處理器進入待機狀態，直到DMA操作完成回傳中斷。每次搬移都會啟用和關閉DMA控制器，確保搬移過程正確。

**Screen shot of wave forms and simulation results**

1. **Syn0**

**一張含有 文字, 螢幕擷取畫面, 設計 的圖片

自動產生的描述**

1. **Syn1**

**一張含有 文字, 螢幕擷取畫面, 陳列, 軟體 的圖片

自動產生的描述**

1. **Syn2**

**一張含有 文字, 螢幕擷取畫面, 多媒體軟體, 陳列 的圖片

自動產生的描述**

1. **Syn3**

**一張含有 文字, 螢幕擷取畫面, 軟體, 作業系統 的圖片

自動產生的描述**

1. **Syn4**

**一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片

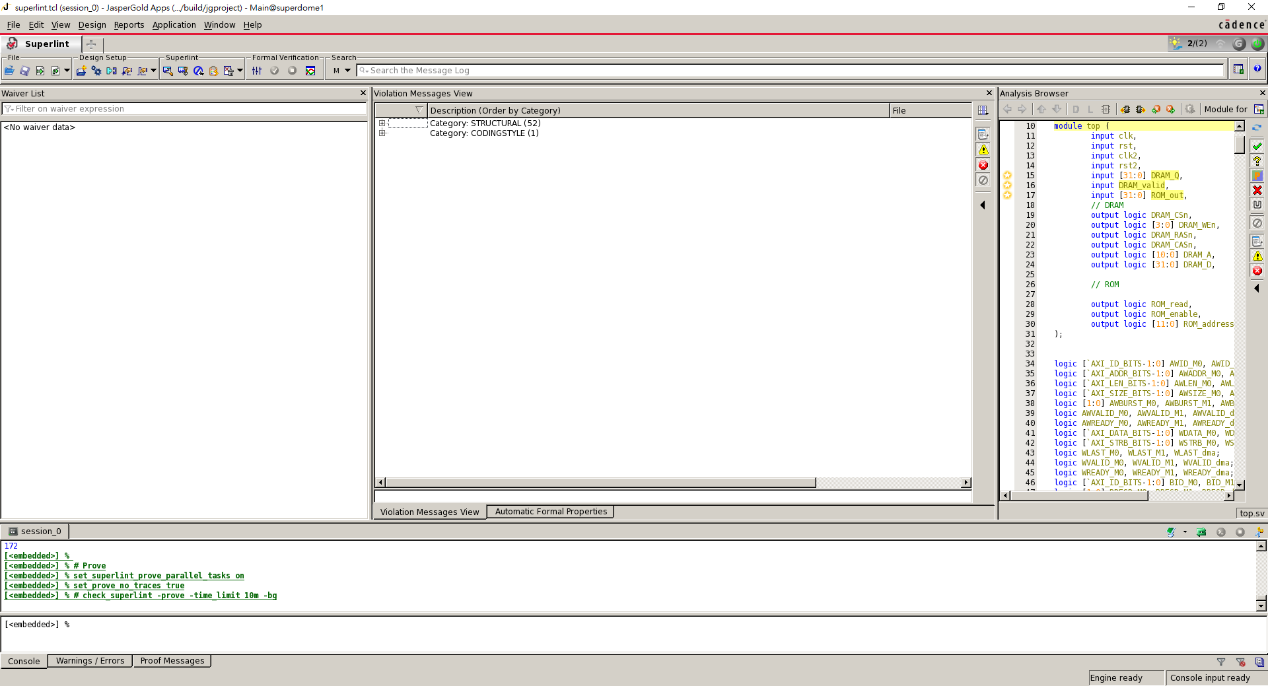
自動產生的描述**

1. **Syn5**

**一張含有 文字, 螢幕擷取畫面, 軟體 的圖片

自動產生的描述**

1. **Superlint**

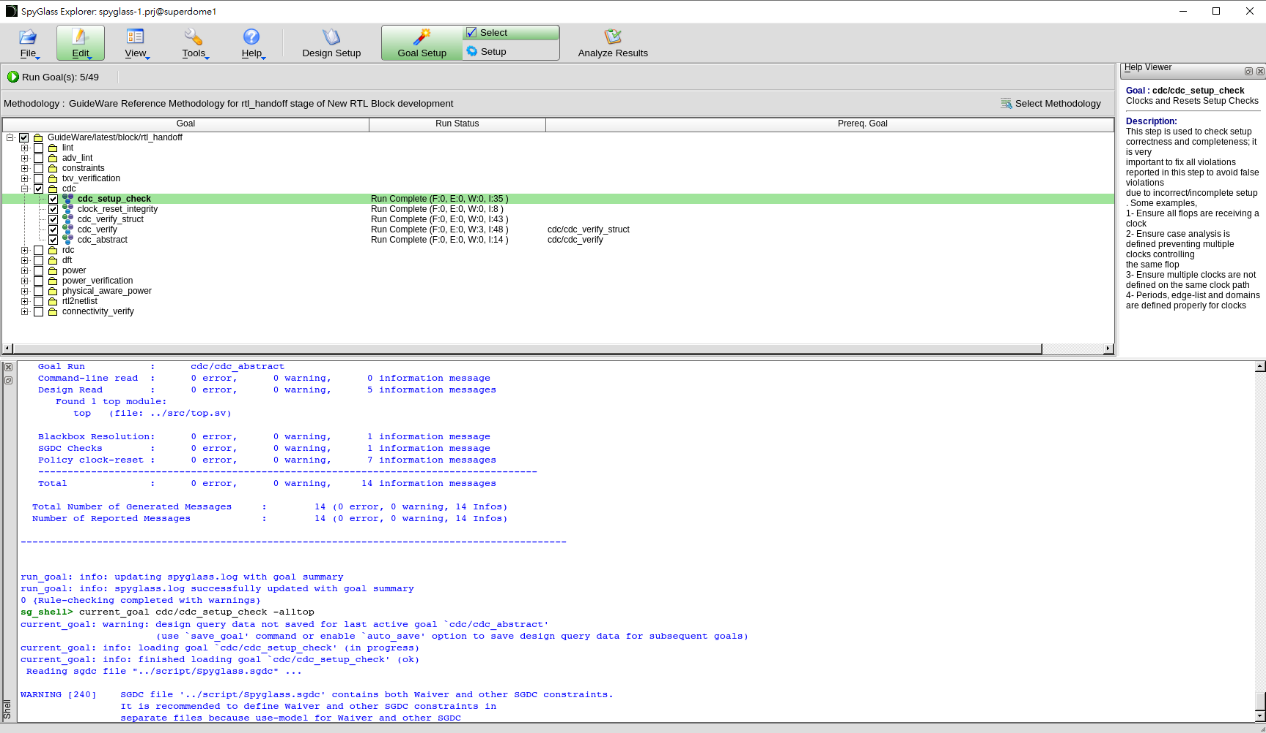
****

1. **Total lines**

**一張含有 文字, 螢幕擷取畫面, 字型, 功能表 的圖片

自動產生的描述**

1. **Spyglass**

****

1. **Setup check**

**一張含有 文字, 螢幕擷取畫面, 陳列, 軟體 的圖片

自動產生的描述**

1. **Clock reset integrity**

**一張含有 文字, 螢幕擷取畫面, 陳列, 軟體 的圖片

自動產生的描述**

1. **Verify struct**

**一張含有 文字, 螢幕擷取畫面, 陳列, 軟體 的圖片

自動產生的描述**

1. **Verify**

**一張含有 文字, 螢幕擷取畫面, 軟體, 陳列 的圖片

自動產生的描述**

1. **Abstract**

**一張含有 文字, 螢幕擷取畫面, 陳列, 軟體 的圖片

自動產生的描述**

1. **DMA**

DMA的運作方式我是透過Burst來達到連續讀取與寫入的做法，進而提升PA。因為DMAEN為1時，地址已拿到，所以Read 和 Write channel在address的handshake可以同時進行，當執行完後，我透過data\_buffer與控制訊號full的暫存器來存取資料及判斷現在應該執行讀取還是寫入。當full為0時，Read data 先執行，RVALID和RREADY同時為1時拿到讀取的資料，將資料存入data\_buffer並把full拉起為1，而因為是Burst的寫法故在RLAST為0的同時將原先的address加4。Read data執行完成後，輪到Write開始執行，判斷data\_buffer的控制訊號full為1時，則將WREADY拉起為1並寫入欲寫入的位址與先前存入data\_buffer的資料，寫完後再將full拉回為0，waddr\_reg的地址加4，使下次輪到Read channel讀取下個address的資料，write channel寫入下個address的資料，這樣做可以減少每次做地址handshake的處理，提升資料搬移的效率。搬移方式如下圖所示。

**一張含有 螢幕擷取畫面, 行 的圖片

自動產生的描述**

1. **WDT**

**一張含有 螢幕擷取畫面, 鮮豔, 電子產品, 行 的圖片

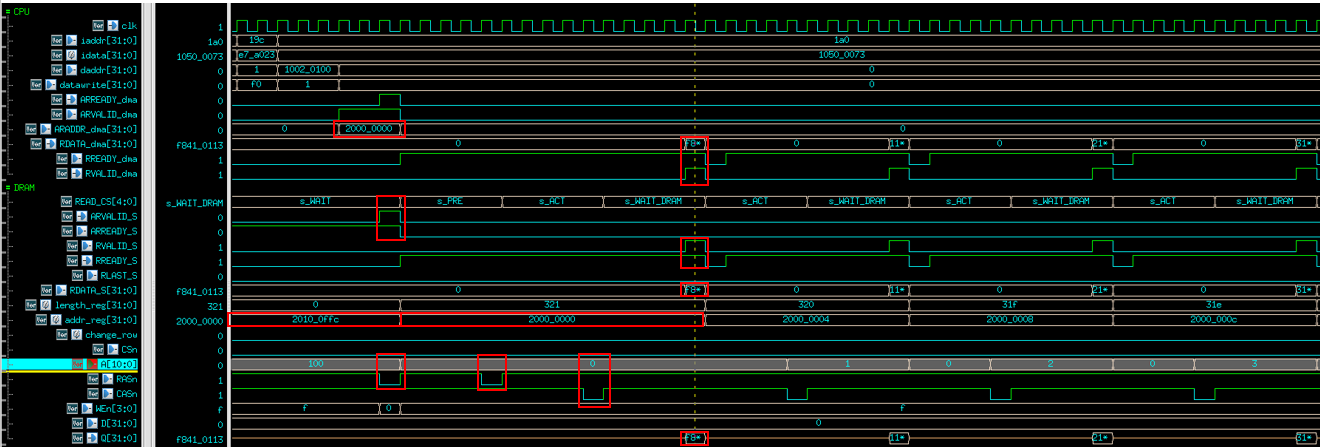
自動產生的描述**

**一張含有 螢幕擷取畫面, 文字, 軟體, 多媒體軟體 的圖片

自動產生的描述**

1. **DRAM**

**Read :** 在DRAM做讀取的時候，若為Address的handshake，則會判斷ARADDR的row位址是否與先前的row address相同，如果不同則會開始進行precharge，將RASn拉為0、Wen為4’h0，A為先前存取的row address，並進入下個狀態s\_PRE，同時更新addr\_reg的暫存器。而因為Precharge需間隔4 cycle才可進行ACT，也就是第5個cycle，故counter從0開始數到4時才將RASn再次拉0，並將A改為寫入先前更新的row address，同時進入下個狀態s\_ACT且將counter歸零。ACT後也須等待4個cycle才能將存取column address，故也是等到counter從0開始數到4時將CASn拉為0與A寫入先前更新的column address，並同時將counter歸零進入下個狀態s\_wait\_DRAM。因為CASn拉為0後仍要等待4個cycle，所以必須停在s\_wait\_DRAM做等待，但是因為回傳的資料會搭配valid訊號，所以可以不用自己使用counter計算，只要等待valid的訊號為1的時候，判斷RREADY是否為1，若同時成立則直接用組合電路將RVALID拉起並WDATA賦與DRAM回傳的Q data。

****

在prog2的連續讀取會遇到讀到一半row需改變的情況，故在s\_WAIT\_DRAM回到s\_ACT時，會判斷此時更新完的address\_reg和先前儲存的row\_addr是否一樣，若不一致則再回到s\_WAIT的狀態，並同時拉起change\_row的flag。因為addr\_reg已經被更新過了，所以透過flag判斷改為將A寫入先前的row\_addr，執行與先前precharge的操作相同，並更新row\_addr。如果一樣時則不須Precharge，每次從ACT開始執行即可。

**一張含有 螢幕擷取畫面, 電子產品, 電路 的圖片

自動產生的描述**

**Write :** 在DRAM做寫入的時候操作與讀取差不多，只差在寫入Column時，Wen要為0，且要寫入的data也要維持住5個cycle，這是在說明裡面沒有，但我去看程式碼才發現的，不過不用特別的處理，只要將欲寫入的data持續暫存到下次更新即可。

**一張含有 螢幕擷取畫面, 電子產品, 電路 的圖片

自動產生的描述**

**Problems to answer**

1. What is the deference between mcycle and timer?

Mcycle的計算內容是時鐘週期，且通常用途為性能的測量，而timer的計算內容是實際時間間隔，用途通常為定時事件、任務排成或延遲操作。而因為計算的內容不同，mcycle更精準於時鐘週期，timer更精準於實際時間。Mcycle不直接支持中斷，只能用以寫入暫存器或記憶體中做為效能紀錄，而Timer可配置中斷的訊號，讓CPU reset。在靈活性方面，mcycle為固定累積cycle，Timer則可以設定為不同的時間間隔。

1. When is mcycle used?

Mcycle的用途主要用於性能監控和程式執行，性能監控包含分析程式效能，計算一段程式碼執行所需的時鐘周期數，衡量其運行效率，以及評估硬體指令效能，在嵌入式系統或硬體設計中，用於比較不同指令或操作的效率。在時間測量上，雖然mcycle計算的是時鐘週期，但通過已知的CPU時鐘頻率可以換算為實際時間。透過以上兩點，可比較不同硬體配置，並辨識出程式耗時較多的部分。

1. What is “Potential Qualifier” in Spyglass?

在程式分析當中，Spyglass檢測到可能被用做條件檢查的訊號或變數，這些條件可能對程式邏輯的行為產生影響。

**Lesson learned**

CSR Instruction 的執行掌握了如何透過控制與狀態暫存器 (CSR) 管理系統中的配置參數與狀態，特別是在中斷管理和系統配置中的應用。這讓我對 CPU 指令執行的底層邏輯有了更深的認識。 DMA 的運作方式深入了解了 DMA（直接記憶體存取）的工作原理，尤其是在減輕 CPU 資源消耗方面的重要性。學習到如何通過 DMA 提高資料傳輸效率，以及與中斷協作的機制。DRAM Controller 的控制方式學會了如何控制 DRAM，以確保資料的正確讀寫與時間同步，並了解其內部的刷新機制 (Refresh) 和記憶體存取的調度策略。這對於設計高效能記憶體子系統有很大幫助。 CPU Interrupt 的處理掌握了中斷處理的基本概念，包括如何設計中斷、保存與恢復上下文，以及中斷處理程式與主程式之間的協作方式。對實現高效且穩定的 CPU 設計非常重要。用 ROM 進行 Booting 學習到如何利用 ROM 存儲開機程式 (Boot Code)，確保系統在通電時能夠正確啟動。理解了 Booting 的過程以及不同階段的初始化步驟。這門課程讓我對 CPU 和相關硬體的設計有了全面的了解，不僅提升了理論基礎，也讓我更加熟悉硬體設計中面臨的實際挑戰，如性能優化、資源分配和可靠性保障。這些知識將對未來的硬體設計和優化工作提供非常寶貴的參考。