VLSI System Design (Graduate Level)

Fall 2021

HOMEWORK II

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: \_\_顧芳宜 \_\_\_\_\_\_\_\_\_\_\_

Student ID: \_\_\_P78101556\_ \_\_\_\_\_\_\_\_\_\_\_

**Summary**

接續上一個5-stage pipelined CPU的實作，此次作業加上以AXI為接口協議的bus作為CPU及記憶體之間資料的傳輸，其中包含2個Masters跟2個Slaves (+一個default slave)。此次作業需要完成：

1. Jaspergold驗證
2. RTL驗證 (包含33個instruction測試、sort algorithm、multiplication& GCD)
3. Gate-Level simulation驗證

**Lesson Learnt**

* 深入學習AXI bus的運作模式和其中需要考慮的問題，比如說如何實現ready handshake和write handshake、等待資料傳遞過程中的stalls。

以下會將實作的AXI bus分為讀和寫進行討論：

* 讀取資料：
  + Read Address Channel (AR)

讀取資料應由master首先發出需要資料的request，而在作業中的master是我們第一個作業實作的CPU，由於CPU 會分別向instruction memory (IM)和Data Memory (DM) 存取資料，我們需要2個masters。Master所發出的request，也就是會傳到AXI bus上的ARADDR(表示需要讀取資料的位置)和ARVALID(表示ARADDR為有效的傳輸)，加上此次transaction的一些資訊，如ID、address、LEN、SIZE、BURST (以ARID\_Mx表示ID，以此類推)。

由於可能同時有兩個masters提出要求存取記憶體，Requests會首先經過Arbiter，決定到底先由哪一個master先進行此次的transaction再被送入Decoder進行解碼，決定是由哪一個slave接收資料。Arbiter 會將決定好的master的ID、address、LEN、SIZE、BURST、VALID傳出到由Decoder根據address選擇的slave(以ARID\_Sx表示ID，以此類推)。

若slave有成功收到ARVALID的訊號，則會傳出READY的訊號(ARREADY\_Sx)表示handshake完成。此時，Master會開始準備接收資料(RREADY拉高)。

* + Read Data Channel (R)

完成AR channel的握手之後，slave會根據ARADDR和ARLEN準備相對應的資料(RDATA)，並同時發出傳遞data的request ，也就是RVALID，並同時發出此次transaction所需要的資訊到AXI bus，包含由ARID\_Sx算出的ID和資料的LAST(RID\_Sx、RLAST\_Sx表示)。AXI bus會根據RID準確將一併將RDATA和RVALID傳給正確的master。而當master在將RREADY拉高的狀態接收到RVALID時，就代表handshake完成，可進行下一筆資料的傳輸，一直到RLAST拉高表示所有資料傳輸的完成。

* 寫入資料：
  + Write Address Channel (AW)

AW channel的運作原理與AR channel相同。由Arbiter決定由哪一個master進行此次的transaction，根據其所傳出的ADDR決定slave (由於此次作業只同意將資料寫入DM，因此arbiter所決定的master永遠都是M1)。當slave收到AWVALID (AWVALID\_Sx)時，便會發出AWREADY (AWREADY\_Mx)的訊號到AXI表示此次握手完成。另外，AXI會將由Master傳出的ID (AWID\_Mx) 傳給slave，以利於等等write response的回傳。

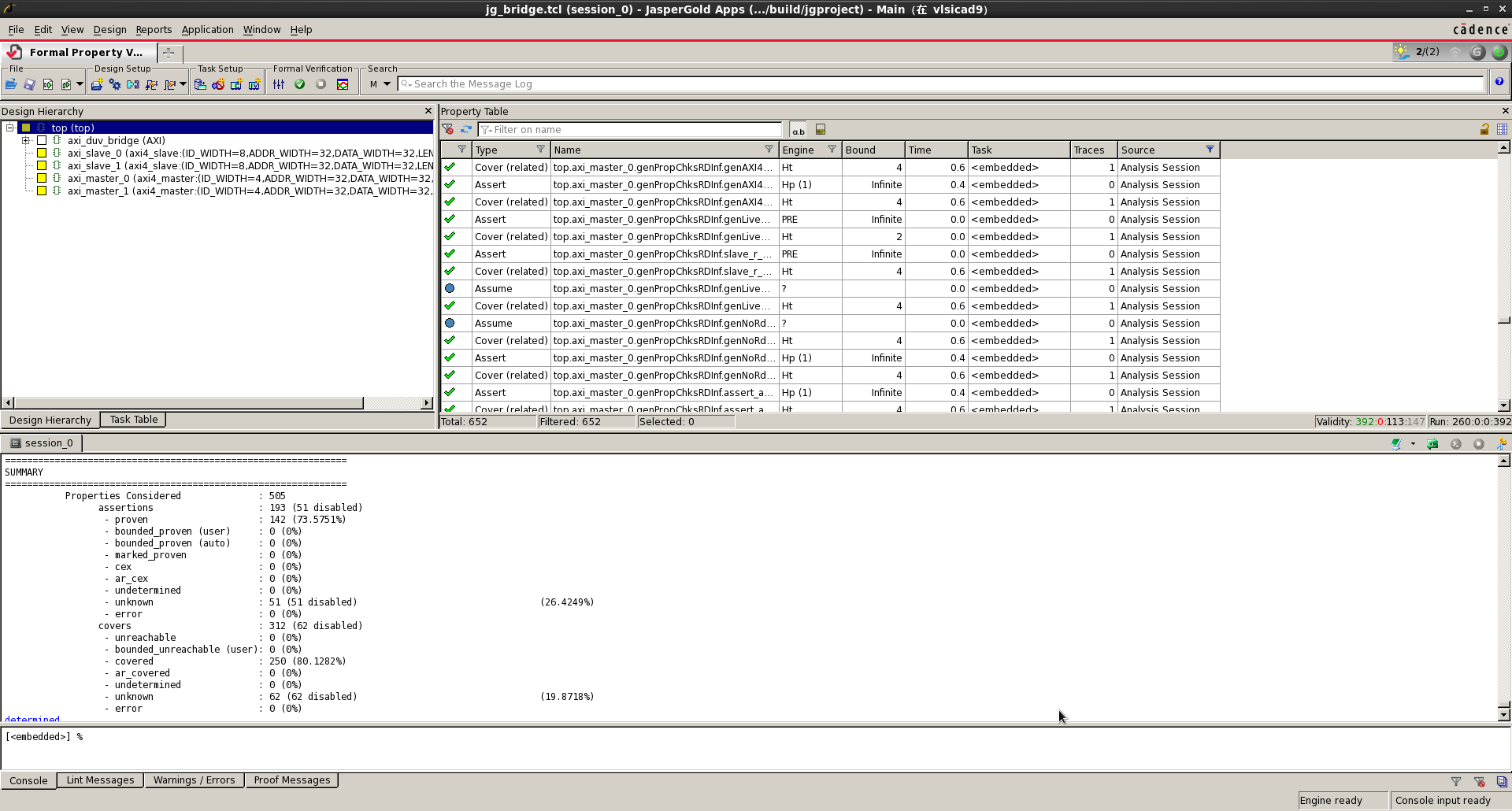
* + Write Data Channel (W)

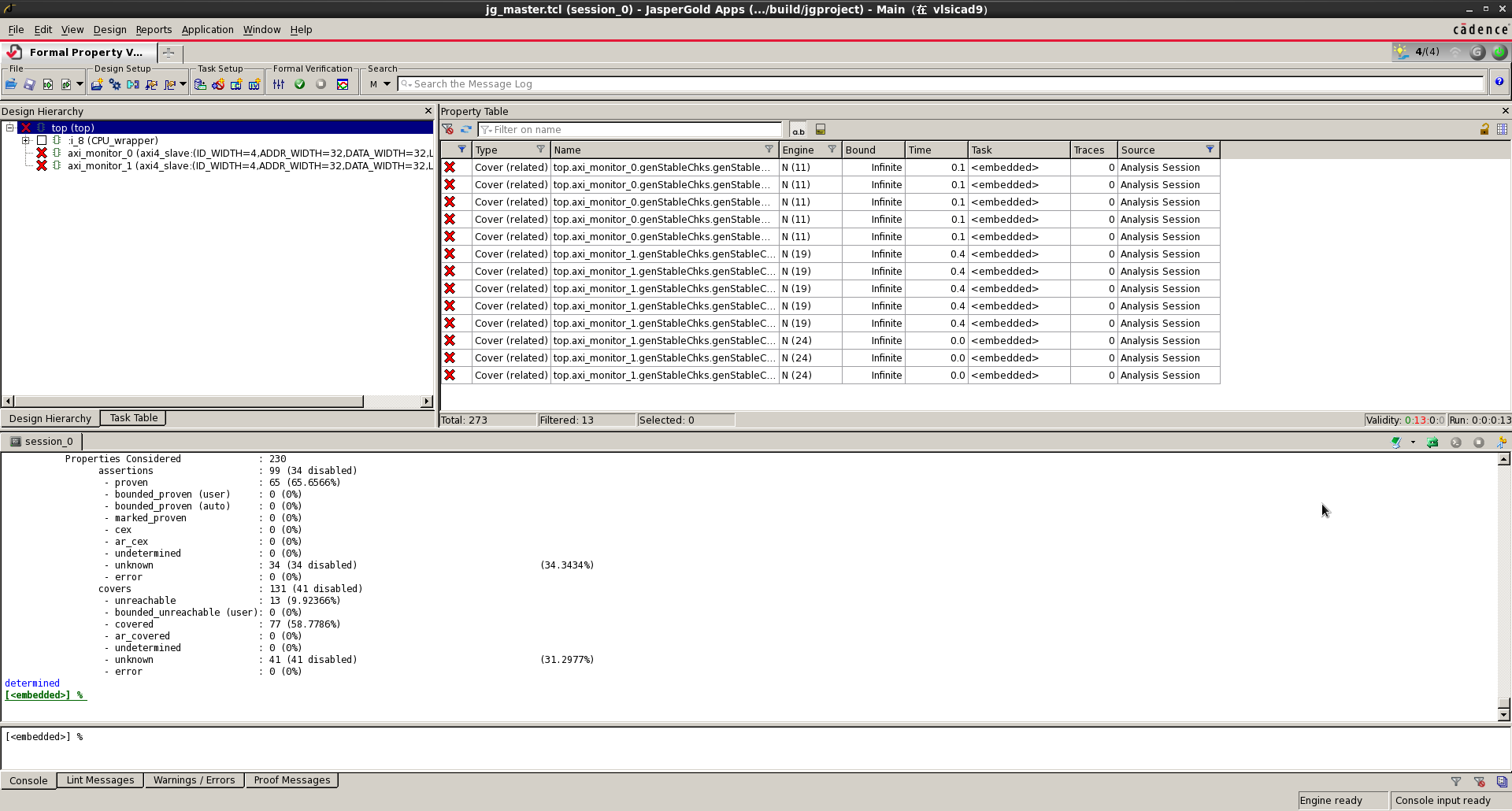
AW握手完成後，master便會開始傳遞要寫入的資料 (WDATA\_Mx) 及資料相關資訊傳到AXI Bus，包含STRB、LAST(以WSTRB\_Mx及WLAST\_Mx表示)，並將WVALID (WVALID\_Mx) 拉高表示有效的傳輸。AXI的W channel會將資料(WDATA\_Sx)及其相關資料傳出(WVALID\_Sx、WSTRB\_Sx及WLAST\_Sx)，並由先前的AWVALID\_Sx決定資料要傳送到的slave。當slave接收到WVALID\_Sx時便會傳出WREADY (WREADY\_Sx)的訊號到AXI再傳給master (WREADY\_Mx)，表示handshake已完成。由於此次作業一次僅寫入一筆資料，因此WLAST設為1。

* + Write Response Channel (B)

W channel 的握手完成後，slave便會傳出BRESP表示此次transaction的狀態到AXI，並且一併傳出BVALID表示此response為有效傳輸，分為以BRESP\_Sx和BVALID\_Sx表示，根據先前的AWID知道回傳資料相對應的master (BID\_Sx)。AXI會根據BID將response的資訊 (BVALID\_Mx) 回傳。若master在BREADY拉高的狀態接收到BVALID，則表示此次握手完成。

**Jaspergold vip Results**

* AXI Bridge  
  
* Master

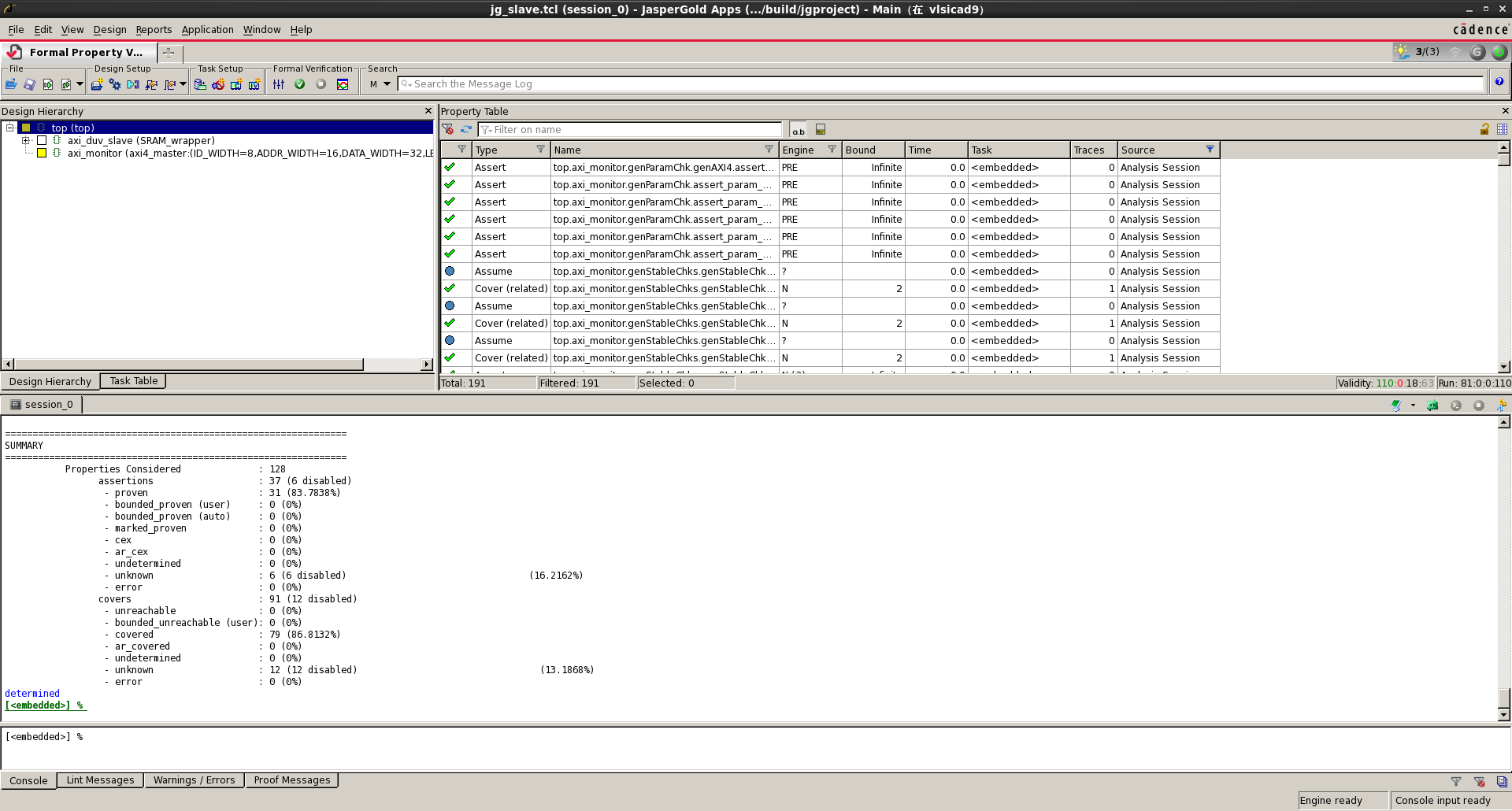


在master中，一共有13個unreachable的情況。這些是由於在AXI bus中B和R channel的設計上的問題。通常在在AR 握手完成時，master便可傳出READY準備接收data，等待VALID的傳入。然而我的設計上將READY和VALID同時拉高，因此會有些狀態無法涵蓋到。

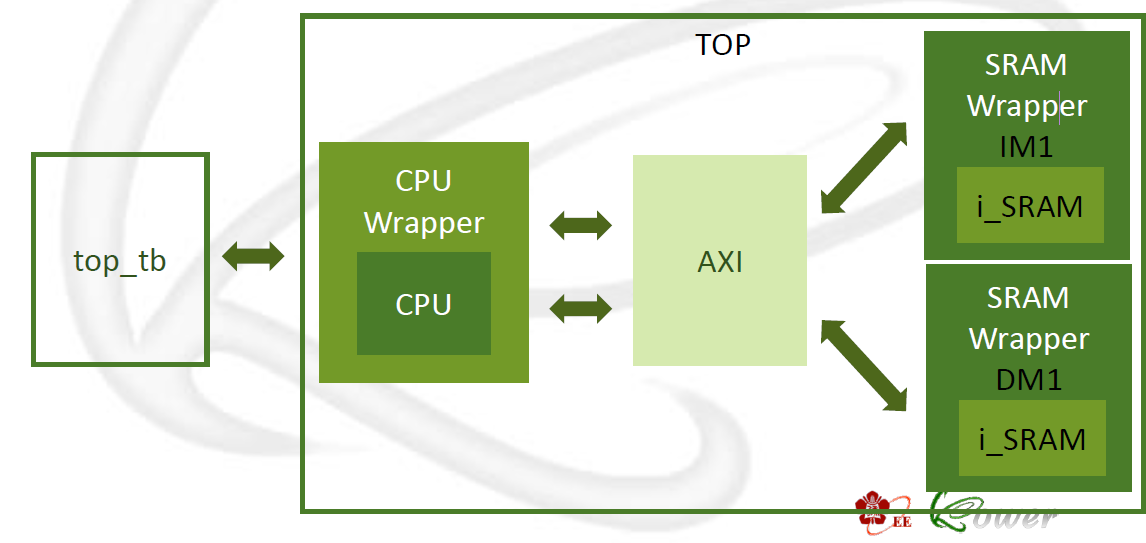
譬如說，jaspergold會驗證是否在RVALID或RLAST拉高時，RREADY仍然為0的狀況，而這種狀況在我的設計中不會發生，因此並無法被涵蓋到，因此會導致RVALID、RRESP、RLAST、RID、RDATA unstable的狀況。

同理，jaspergold會驗證是否在R=BVALID或BLAST拉高時，BREADY仍然為0的狀況，而這種狀況在我的設計中不會發生，因此並無法被涵蓋到，因此會導致BVALID、BRESP、BLAST、BID、BDATA unstable的狀況。

* Slave



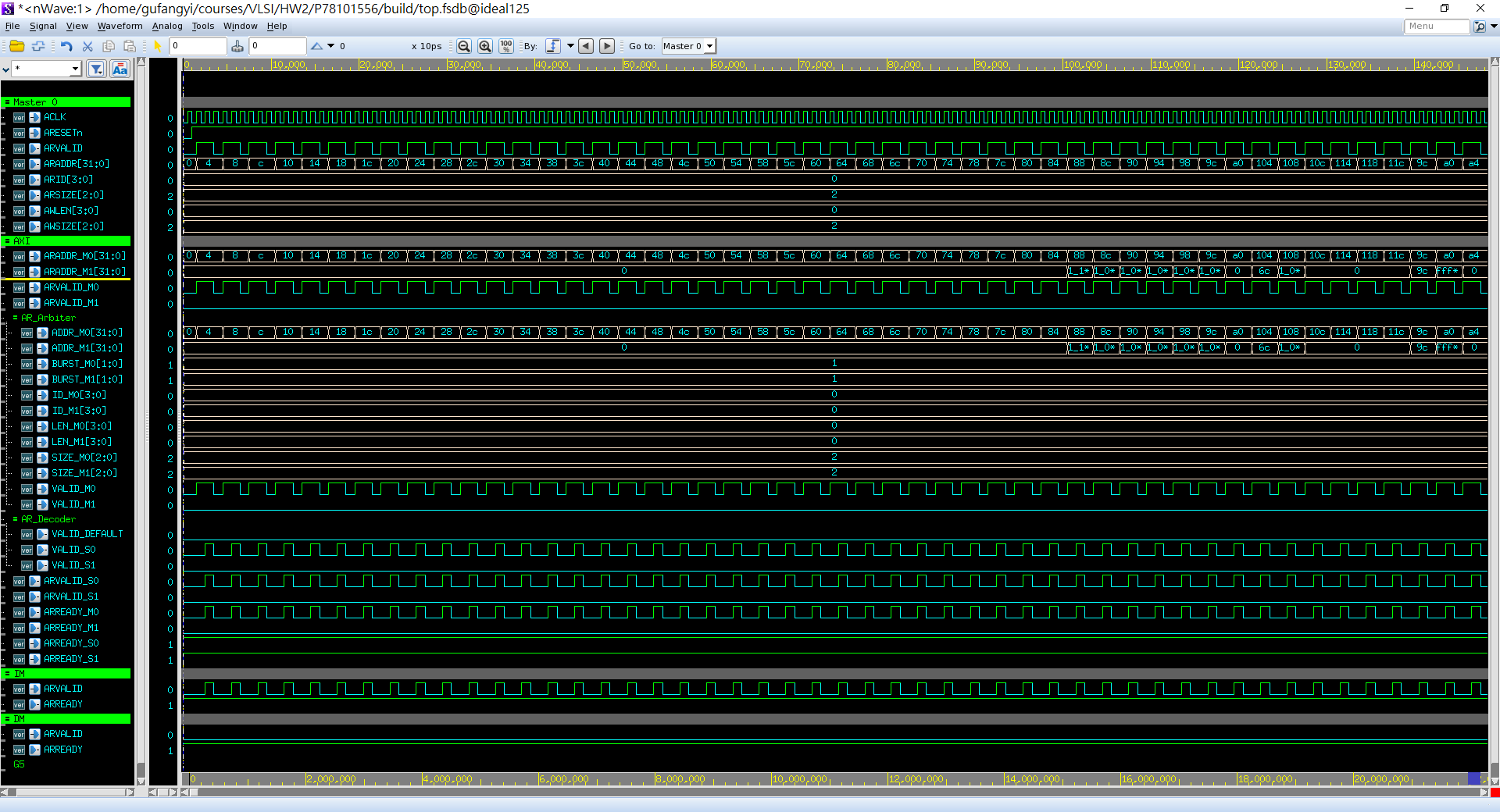
**Block Diagram**



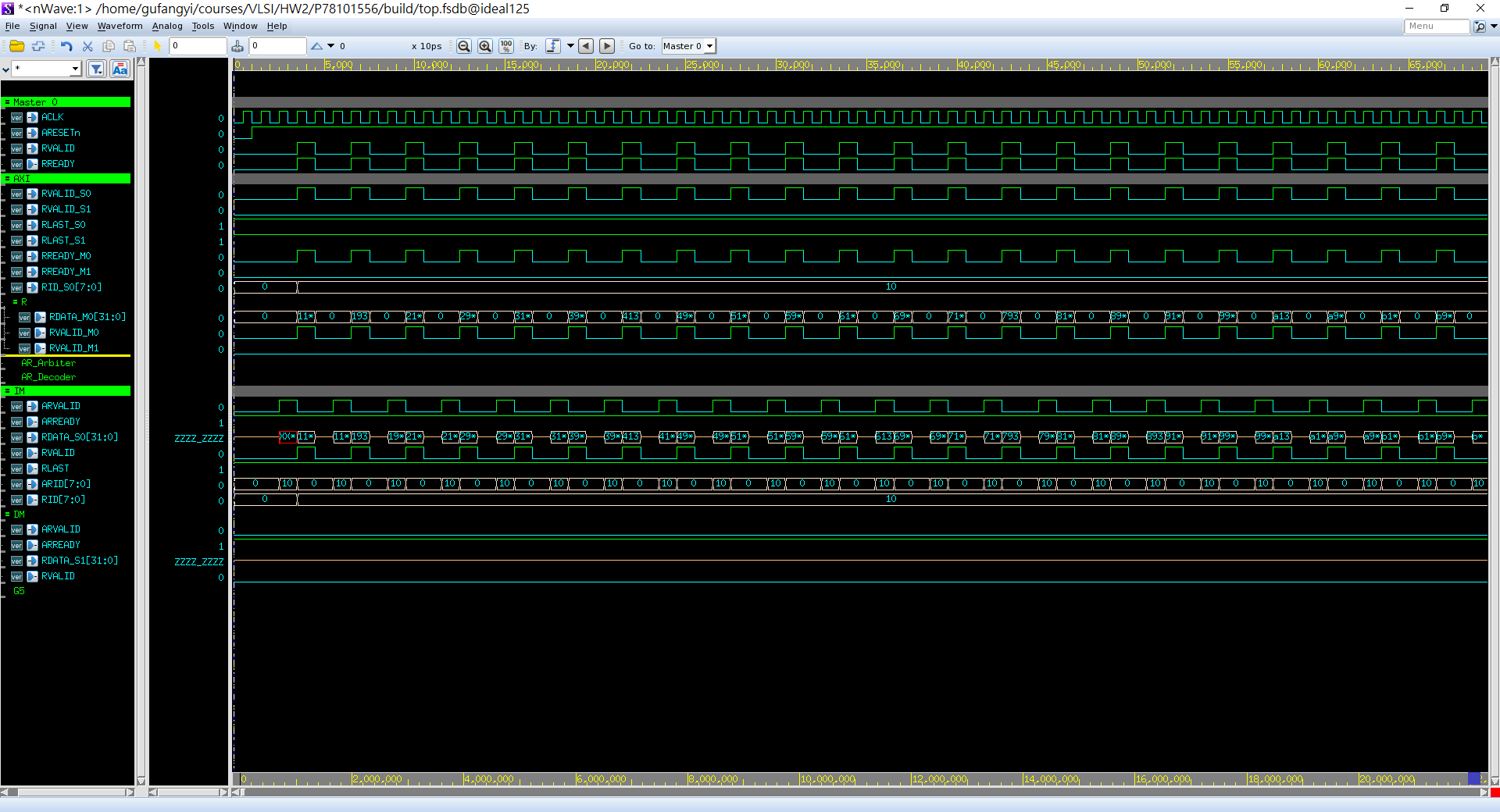
以上架構圖擷取至Homework II 說明文件。CPU Wrapper中包含上面所述的兩個Masters，透過中間的AXI bridge 與 包含在SRAM Wrapper中的兩個Slaves進行transactions。

**Waveform Verification**

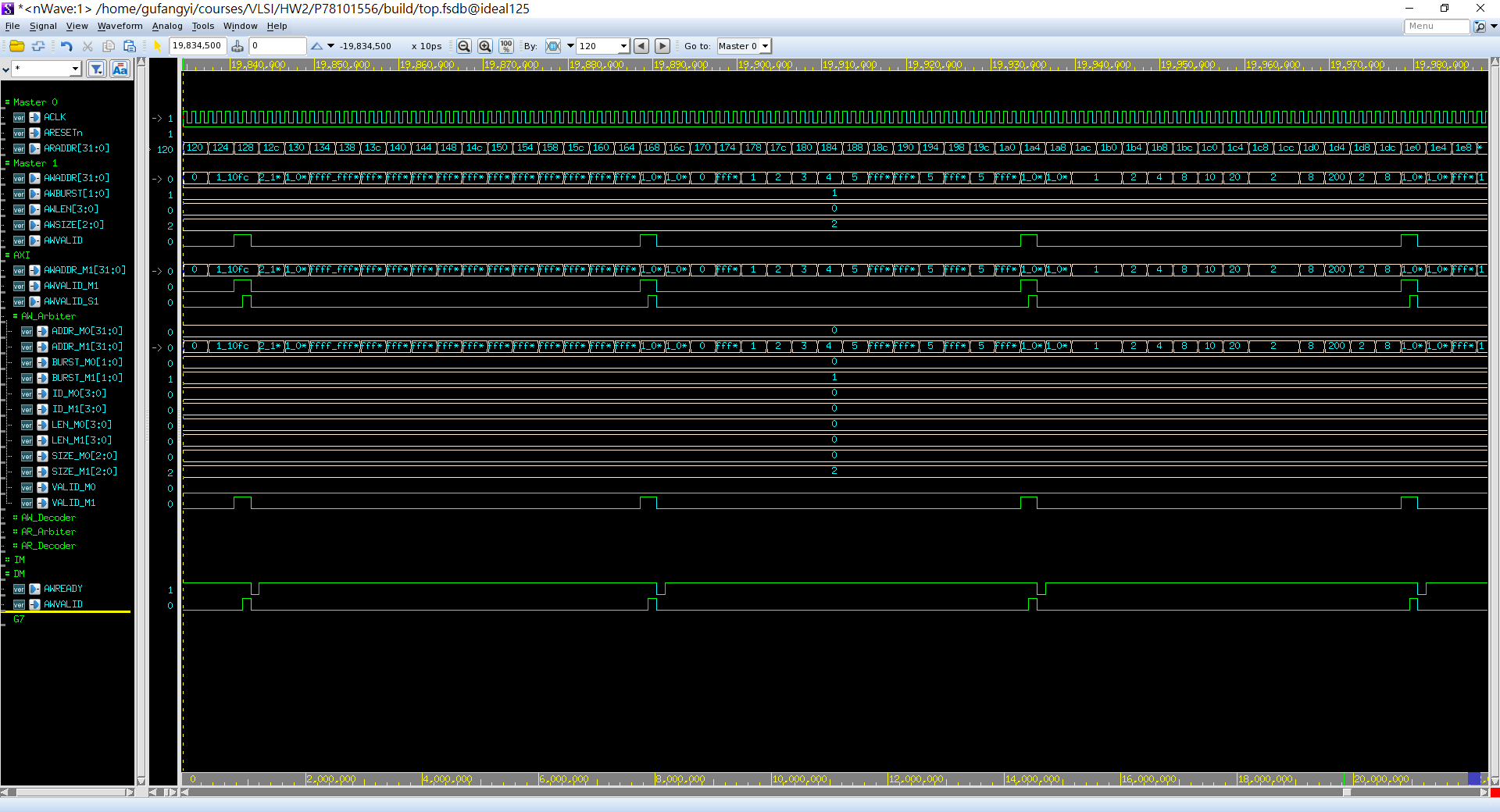
如上方AXI 的說明，這邊的waveform verification會用讀寫兩個部份表示。

* 讀取資料：
  + Read Address Channel (AR)

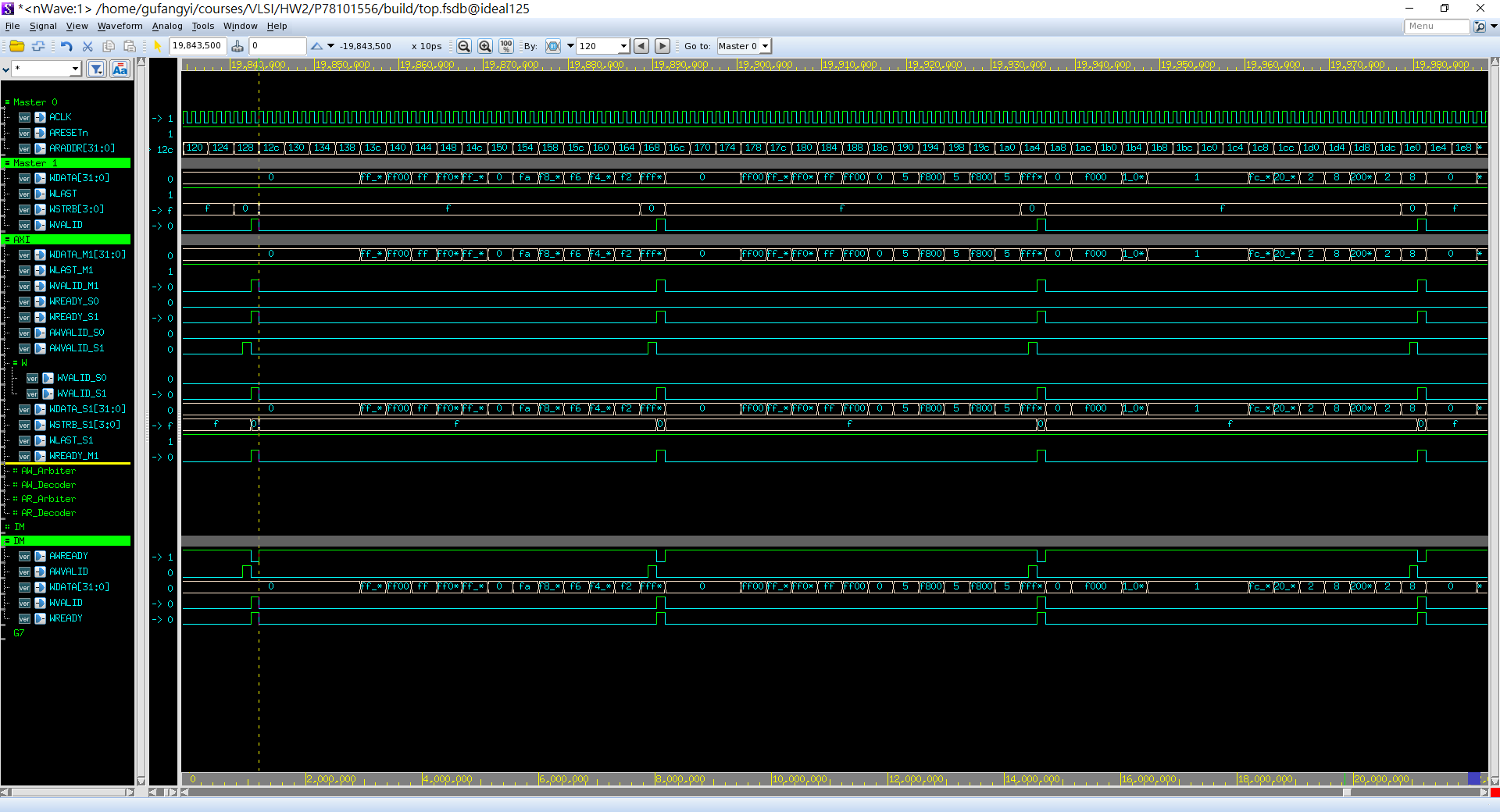
這是程式起始的波形圖，用從IM讀取instruction來解釋AR channel的訊號。首先是由Master 0 傳出ARADDR\_M0和ARVALID\_M0到AXI中，經過AR\_Arbiter決定是否使此master進行transaction後， AR\_Decoder會透過ARADDR\_M0決定傳遞終點，將ARVALID\_M0訊號傳給相對應的Slave，輸出ARVALID\_S0。由於AXI specification沒有強制規定ARREADY合ARVALID的相依性，我將IM的ARREADY一直設為拉高的狀態。當IM接收到ARVALID訊號時，握手完成。

* + Read Data Channel (R)

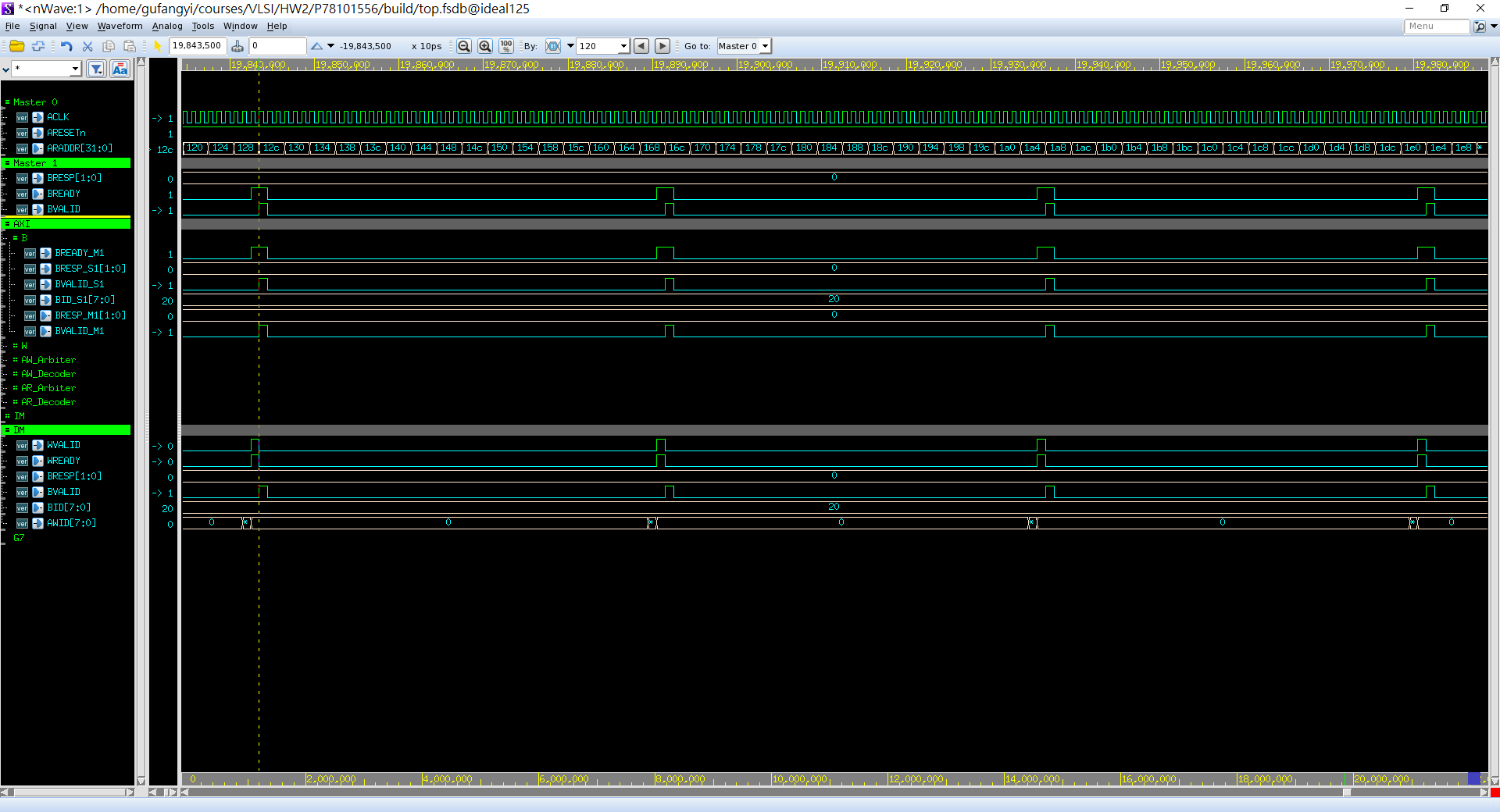
AR 握手完成後，IM 會將RDATA\_S0、RVALID和由ARID決定的RID傳到AXI，而RLAST為1。AXI接收到RLAST\_S0、RVALID\_S0等訊號後，會由RID\_S0決定會傳的master，並將RVALID\_M0傳出。Master此時會接收到RVALID，並將RREADY拉起表示握手成功，這個RREADY\_M0會傳到AXI上。

* 寫入資料：
  + Write Address Channel (AW)

這邊以將資料寫入DM作為例子。首先是由Master 1 傳出AWADDR\_M1和AWVALID\_M1到AXI中，經過AW\_Arbiter決定是否使此master進行transaction後， AW\_Decoder會透過AWADDR\_M1決定傳遞終點，將AWVALID\_M1訊號傳給相對應的Slave，輸出AWVALID\_S1。由於AXI specification沒有強制規定AWREADY和AWVALID的相依性，我將DM的AWREADY一直設為拉高的狀態直到接收到VALID。當DM接收到AWVALID訊號時，握手完成。

* + Write Data Channel (W)

W握手完成後，Master1便會開始傳遞要寫入的資料 WDATA\_M1 及資料相關資訊傳到AXI Bus，包含WSTRB\_M1及WLAST\_M1，並將WVALID\_M1 拉高表示有效的傳輸。AXI的W channel會將資料以WDATA\_S1及WVALID\_S1等相關資料傳出，並由先前的AWVALID\_S1決定資料要傳送到的slave。當slave接收到WVALID\_S1時便會傳出WREADY\_S1的訊號到AXI再傳給master (WREADY\_M1)，表示handshake已完成。由於此次作業一次僅寫入一筆資料，因此WLAST設為1。

* + Write Response Channel (B)

W channel 的握手完成後，slave便會傳出BRESP表示此次transaction的狀態到AXI，並且一併傳出BVALID表示此response為有效傳輸，分為以BRESP\_S1和BVALID\_S1傳到AXI，並根據先前的AWID知道回傳資料相對應的BID。AXI會根據BID\_S1 將BVALID\_M1 回傳。Master在得知W握手成功後就會將BREADY拉起，在BREADY拉高的狀態接收到BVALID時，則表示此次握手完成。