VLSI System Design (Graduate Level)

Fall 2023

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: 陳俐蓉

Student ID: N26120113

Outline

[一、 Summary 3](#_Toc149525761)

[二、 Design description 3](#_Toc149525762)

[三、 Verification waveform 12](#_Toc149525763)

[四、 VIP 14](#_Toc149525764)

[五、 Synthesize result 16](#_Toc149525765)

[六、 Problems encountered and lessons learned 22](#_Toc149525766)

1. Summary

在Master(CPU)和Slave(SRAM)之間加上Bridge(AXI)，Bridge負責決定多個Master讀寫Slave的優先權，用來確保多個Master想使用同一個Slave資源時，資料可以正確的傳輸。為了使Master和Slave可以跟Bridge互相溝通，Master和Slave要加上wrapper作為和Bridge溝通的介面。

1. Design description

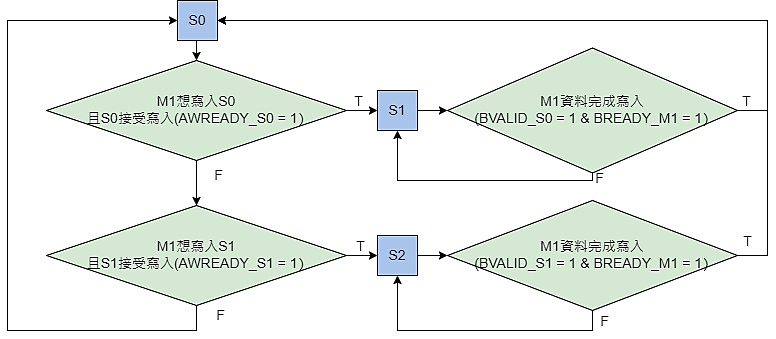
* **AXI (Bridge)**
* 主要負責以下三項任務:

1. 判斷master0(instruction) / master1(data)想要讀寫哪一個slave(IM / DM)，如果master0和master1同時要求讀取同一個slave，則會讓master1優先讀取。
2. 利用狀態機判斷目前的讀寫行為進行到哪個階段(Address phase / Data phase)，以決定目前master要讀寫的slave是否可以接受讀寫。
3. 根據讀寫狀態用MUX分配master要接受哪個slave的訊號，以及slave要接受哪個master的訊號，一旦slave接受某一個master的讀寫，就會一直維持接受該master的訊號直到該次讀寫行為結束，master同理。

* 狀態機

**<========================寫=======================>**

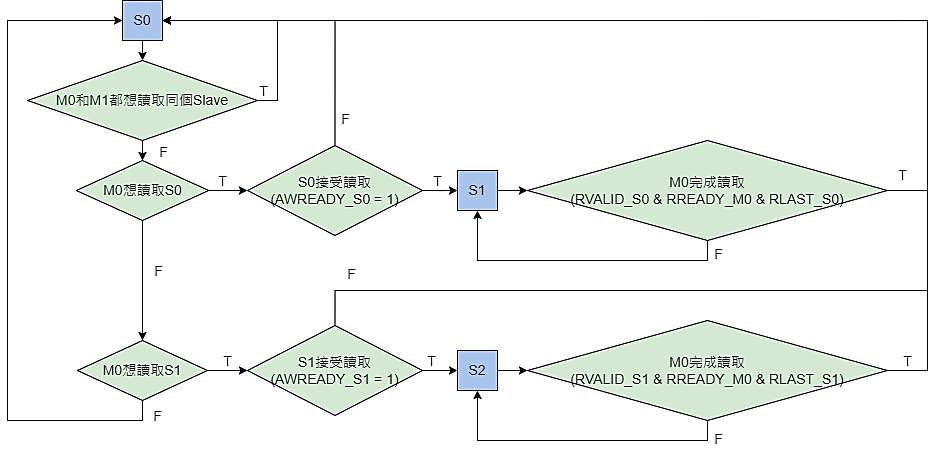
Master1:



|  |  |
| --- | --- |
| M1\_W\_State | Description |
| S0  (AW) | Master1未發出寫要求 / Master1已發出寫要求但Slave尚未接受。 |
| S1  (W/B) | Slave0已接受寫入，Master1正在對Slave0寫入資料且尚未完成。 |
| S2  (W/B) | Slave1已接受寫入，Master1正在對Slave1寫入資料且尚未完成。 |

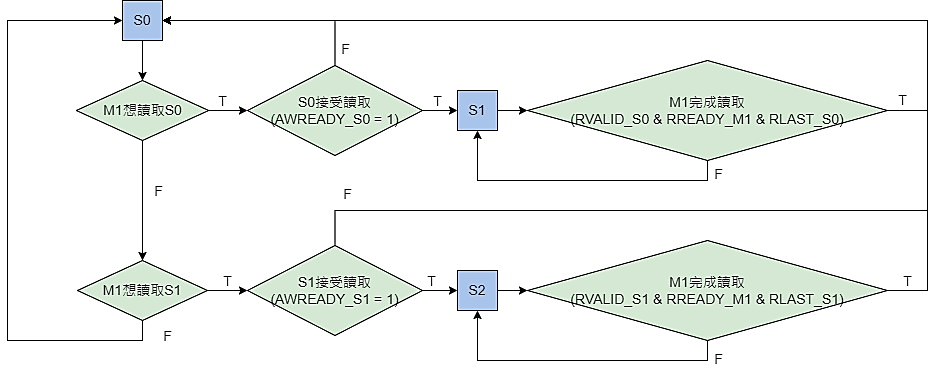
**<========================讀=======================>**

Master0:



|  |  |
| --- | --- |
| M0\_R\_State | Description |
| S0  (AR) | Master0未發出讀要求 / Master 0已發出讀要求但Slave尚未接受。 |
| S1  (R) | Slave0已接受讀取，Master0正在對Slave0讀取資料且尚未完成。 |
| S2  (R) | Slave1已接受讀取，Master0正在對Slave1讀取資料且尚未完成。 |

Master1:

****

|  |  |
| --- | --- |
| M1\_R\_State | Description |
| S0  (AR) | Master1未發出讀要求 / Master 1已發出讀要求但Slave尚未接受。 |
| S1  (R) | Slave0已接受讀取，Master1正在對Slave0讀取資料且尚未完成。 |
| S2  (R) | Slave1已接受讀取，Master1正在對Slave1讀取資料且尚未完成。 |

* Bus訊號分配

**<========================寫=======================>**

|  |  |  |
| --- | --- | --- |
| 訊號接收者 | 訊號發送者 | 條件 |
| Master1 | Slave0 | 1. AW phase(S0)時Master1想寫入Slave0 2. W/B phase(S1) |
| Slave0 | Master1 |
| Master1 | Slave1 | 1. AW phase(S0)時Master1想寫入Slave1 2. W/B phase(S2) |
| Slave1 | Master1 |

**<========================讀=======================>**

|  |  |  |
| --- | --- | --- |
| 訊號接收者 | 訊號發送者 | 條件 |
| Master0 | Slave0 | 1. AR phase(S0)時Master0想讀取Slave0，且Slave0接受讀取 2. R phase(S1) |
| Slave0 | Master0 |
| Master0 | Slave1 | 1. AR phase(S0)時Master0想讀取Slave1，且Slave1接受讀取 2. R phase(S2) |
| Slave1 | Master0 |
| Master1 | Slave0 | 1. AR phase(S0)時Master1想讀取Slave0，且Slave0接受讀取 2. R phase(S1) |
| Slave0 | Master1 |
| Master1 | Slave1 | 1. AR phase(S0)時Master1想讀取Slave1，且Slave1接受讀取 2. R phase(S2) |
| Slave1 | Master1 |

* **CPU wrapper**

CPU wrapper為Master(CPU)與Bridge(AXI)溝通的介面，負責產生和接收與Bridge(AXI)溝通的訊號。當VALID訊號拉起但Slave尚未READY時，CPU wrapper需將Slave所需的資料暫存起來。包含Master0和Master1兩個Master，Master0和Master1皆會進行讀取行為，只有Master1會進行寫入行為。

**<========================寫=======================>**

* Master狀態機



AW : Master不需寫入 / 正在傳輸write address

W : Master正在傳輸write data

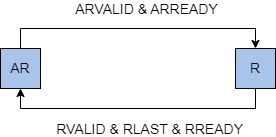
B : Slave正在傳輸write response

* Master1 Wrapper訊號

|  |  |
| --- | --- |
| Signal | Description |
| Address channel | |
| AWID | 維持4'd1 |
| AWLEN | 維持4'd0 (burst length = 1，每次只寫一筆資料) |
| AWSIZE | 維持3'b010 (burst size = 1 word，每次寫資料的大小為1 word) |
| AWBURST | INCR type |
| AWVALID | AW phase時為CPU的ID\_EX \_MemWrite，W/B phase或reset時為low |
| AWADDR | CPU EX stage的ALU result |
| Data channel | |
| WVALID | W phase時為CPU的EX\_MEM\_MemWrite，AW/B phase或reset時為low |
| WDATA | 同筆資料在WVALID拉起後需要維持穩定直到WREADY拉起 |
| WSTRB | 規則同WDATA |
| WLAST | WVALID拉起且資料傳輸的次數達AWLEN時為high，表示正在寫入最後一筆資料 |
| Response channel | |
| BREADY | W/B phase時為high，AW phase為low |

**<========================讀=======================>**

* Master狀態機



AR : Master不需讀取 / 正在傳輸read address

R : Slave正在傳輸read data

* Master0 Wrapper訊號

|  |  |
| --- | --- |
| Signal | Description |
| Address channel | |
| ARID | 維持4'd0 |
| ARLEN | 維持4'd0 (burst length = 1) |
| ARSIZE | 維持3'b010 (burst size = 1 word) |
| ARBURST | INCR type |
| ARVALID | AR phase時為high，R phase為low |
| ARADDR | CPU的指令PC |
| Data channel | |
| RREADY | 維持high |

* Master1 Wrapper訊號

|  |  |
| --- | --- |
| Signal | Description |
| Address channel | |
| ARID | 維持4'd1 |
| ARLEN | 維持4'd0 (burst length = 1) |
| ARSIZE | 維持3'b010 (burst size = 1 word) |
| ARBURST | INCR type |
| ARVALID | AR phase時為ID\_EX \_MemRead，R phase為low |
| ARADDR | CPU EX stage的ALU result |
| Data channel | |
| RREADY | 同CPU的EX\_MEM\_MemRead |

**<========================CPU=======================>**

由於加上AXI後Slave的讀寫動作至少要在2個cycle以上才能完成，所以當下次有Master想讀取同個Slave時，Slave可能尚未做完上次的讀寫，此時Master就會停住等Slave接受讀寫，故CPU有多加IM\_busy和DM\_busy兩條訊號，若busy訊號拉起表示此時memory無法接受新的讀寫要求，指令和PC都要暫停。

|  |  |
| --- | --- |
| Signal | Condition |
| IM\_busy | 1. Master0發出讀取要求(ARVALID = 1)，但Slave尚未接受(ARREADY = 0) 2. Master0處於R phase，資料尚未讀取完成 |
| DM\_busy | 1. Master1發出讀取要求(ARVALID = 1)，但Slave尚未接受(ARREADY = 0) 2. Master1發出寫入要求(AWVALID = 1)，但Slave尚未接受(AWREADY = 0) 3. Master1處於W phase，資料尚未寫入完成 |

* **SRAM wrapper**

SRAM wrapper為Slave(IM / DM)與Bridge(AXI)溝通的介面，當VALID訊號拉起但Slave尚未READY時，Salve不會接受新的讀寫訊號。SRAM wrapper的讀寫狀態機和CPU wrapper類似，寫狀態分為AW、W、B，讀狀態分為AR、R。

**<========================寫=======================>**

* Slave Wrapper訊號

|  |  |
| --- | --- |
| Signal | Description |
| Address channel | |
| AWREADY | 可分為3種情況:   1. reset之後: AWREADY=1 2. W phase / B phase / R phase(亦即Slave正在忙碌): AWREADY=0 3. 其他: AWREADY=1 |
| Data channel | |
| WREADY | 可分為3種情況:   1. reset之後: WREADY=1 2. B phase: WREADY=0   其他: WREADY=1 |
| Response channel | |
| BID | 只有當AWVALID和AWREADY皆拉起時BID才可更新成新的AWID |
| BRESP | 維持OKAY |
| BVALID | B phase時維持high，其他時候為low |

**<========================讀=======================>**

* Slave Wrapper訊號

|  |  |
| --- | --- |
| Signal | Description |
| Address channel | |
| ARREADY | 規則同AWREADY，因為一旦slave開始進行讀寫，直到該次讀寫結束前將不再接受其他讀寫要求 |
| Data channel | |
| RID | 只有當ARVALID和ARREADY皆拉起時RID才可更新為新的ARID |
| RVALID | R phase時會拉起，需維持high直到RREADY拉起 |
| RDATA | 同筆資料在RVALID拉起後需維持穩定直到RREADY拉起 |
| RRESP | 維持OKAY |
| RLAST | RVALID拉起且資料傳輸的次數達ARLEN時為high，表示正在讀取最後一筆資料 |

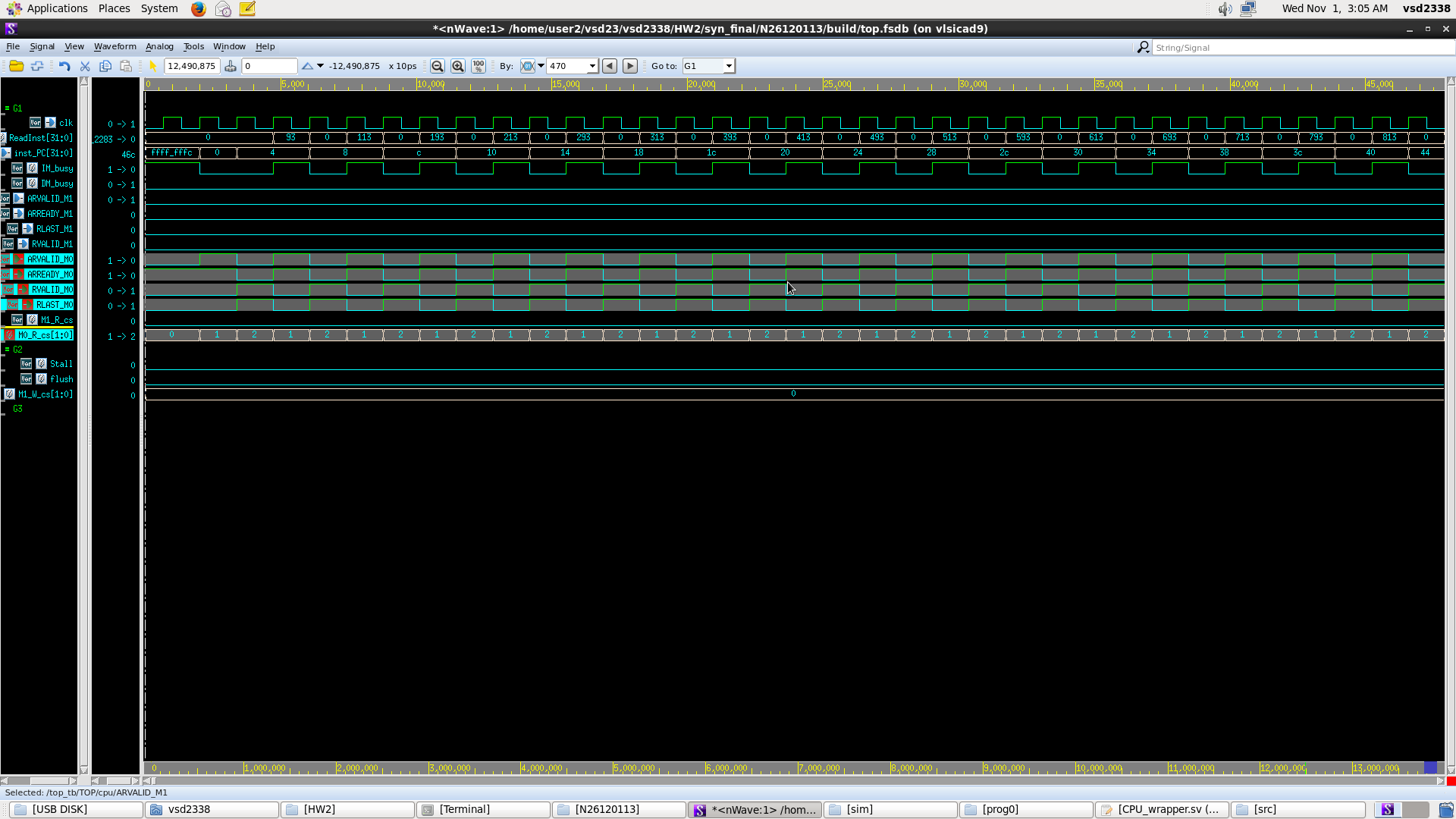
1. Verification waveform

* Master0 read slave0 (fetch instruction)

AR

R

AR



如果ARVALID和ARREADY在AR phase同時為high，則下個cycle會進入R phase，讀完指令後下個cycle又會回到AR phase。

(EX stage)

(ID stage)

(MEM stage)

* Master1 read slave0 (load)

AR

AR

R



AR phase時第一個cycle ARVALID拉起時slave0(此時視為DM)尚未ready，DM\_busy拉起，Master1需停住等待，因為此時Master0正在讀取slave0，下個cycle Master0讀取完成，此時換Master1開始讀取slave0，在Master1尚未完成讀取slave0前，Master0不可讀取slave0，因此可以看到在Master1讀取slave0時IM\_busy會維持high，Master0需暫停PC。

* Master1 write slave1 (store)

AW

(WB stage)

(MEM stage)

(ID stage)

(EX stage)

B

W

AW

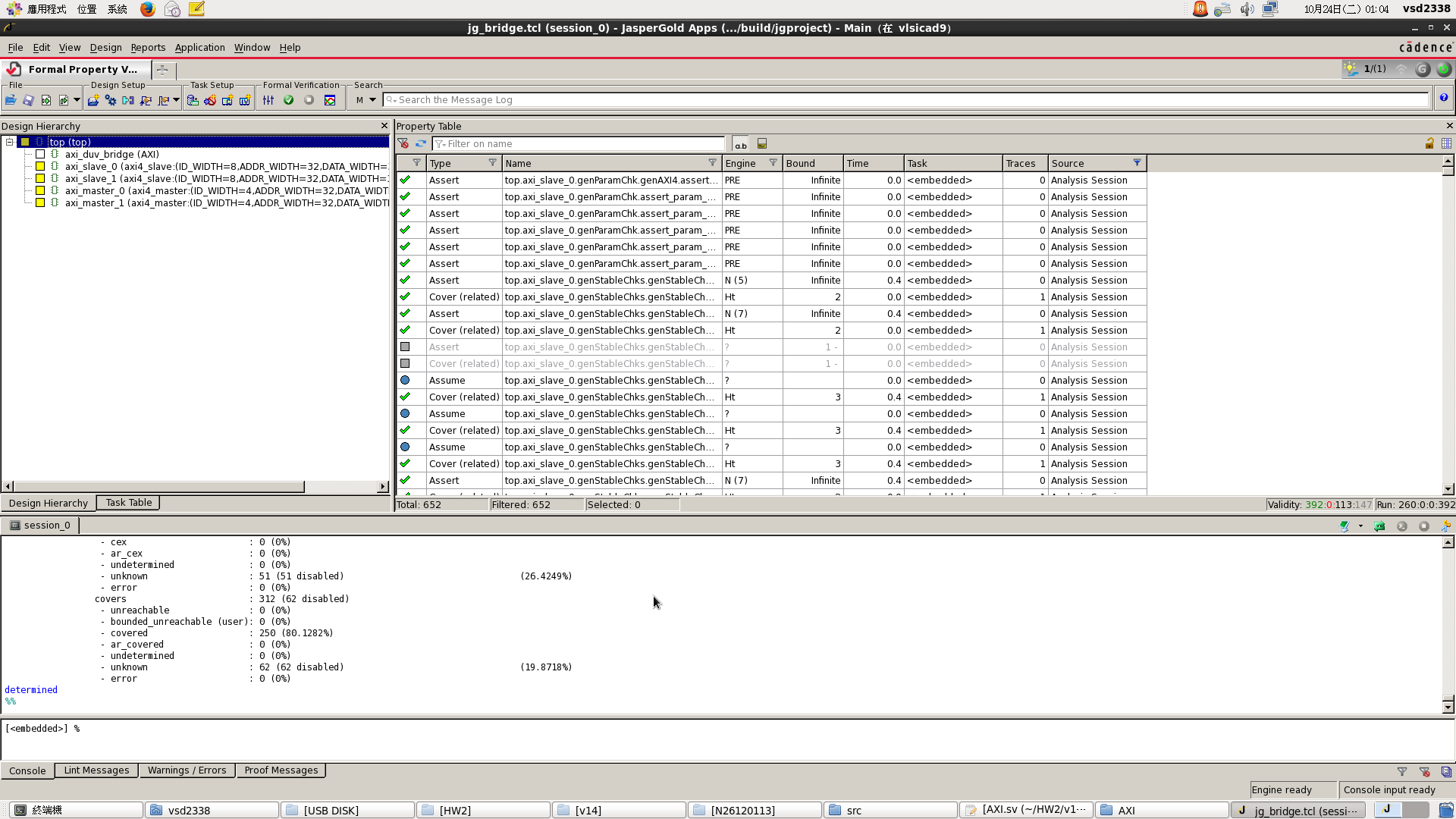


如果AWVALID和AWREADY在AW phase同時為high，則下個cycle會進入W phase，W phase對應的是store指令的MEM stage，W phase結束後下個cycle會進入B phase，B phase結束後會再回到AW phase。

1. VIP

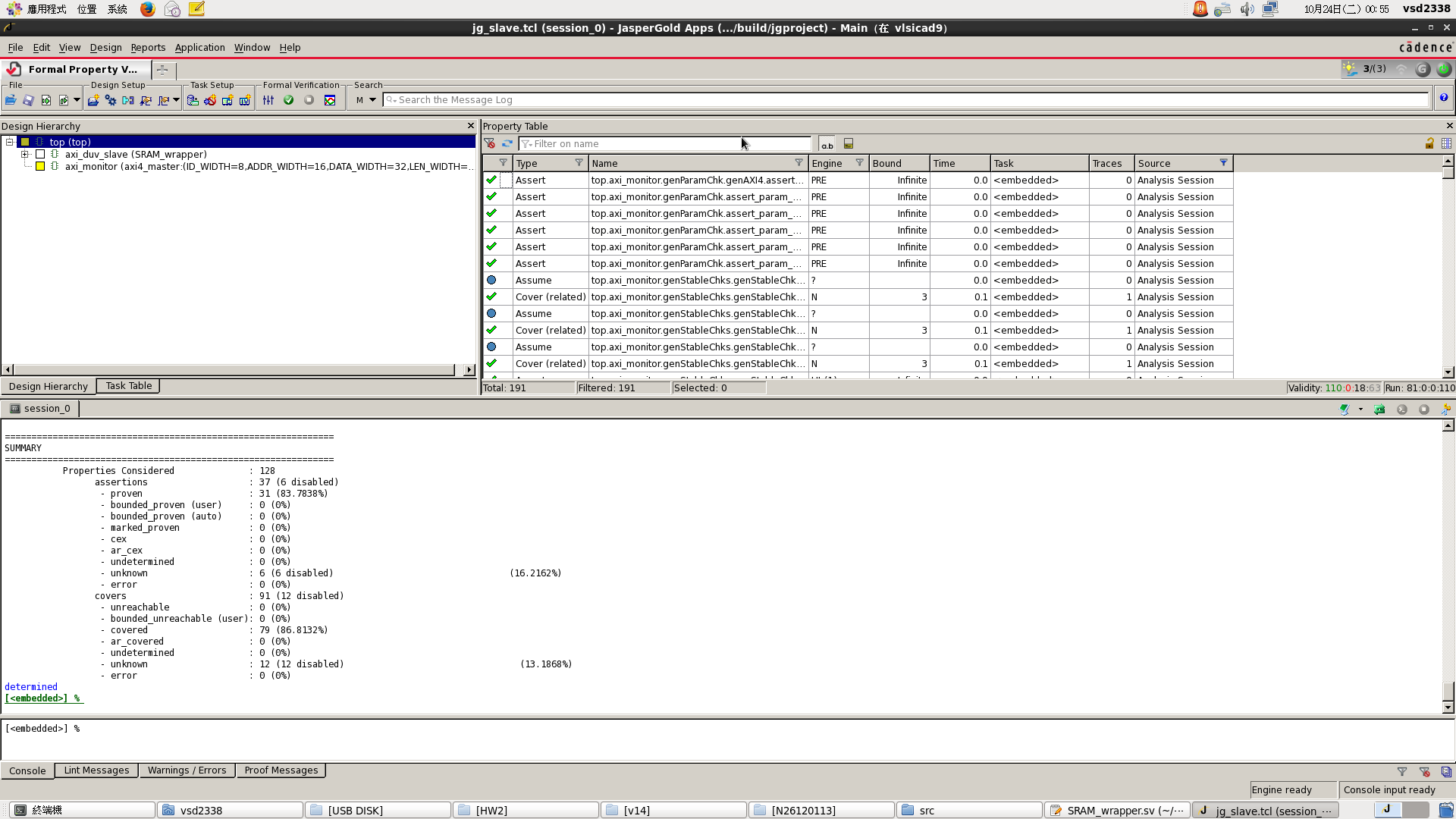
* **AXI(Bridge)**

Assertion all proven

****

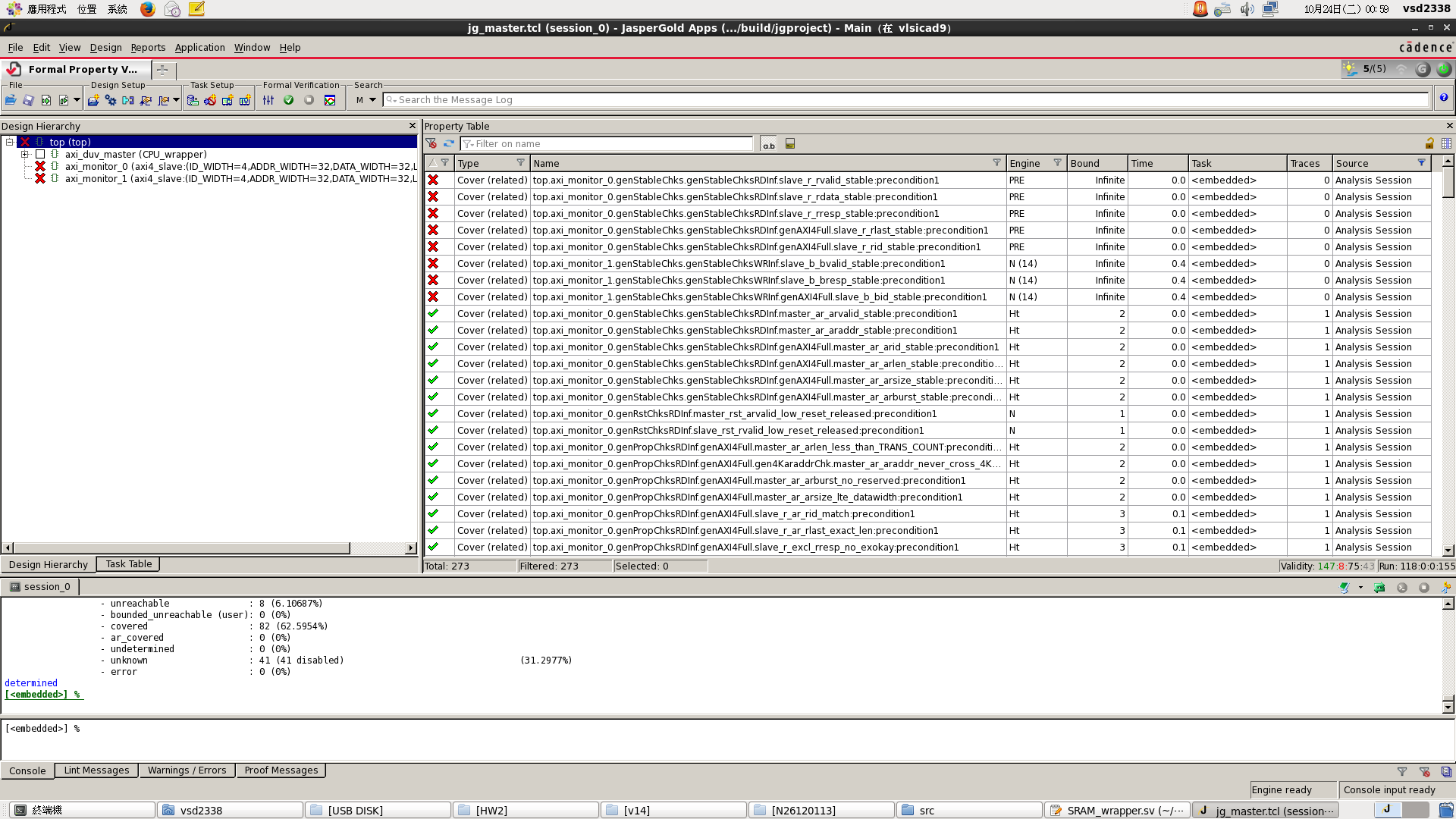
* **SRAM wrapper**

Assertion all proven

****

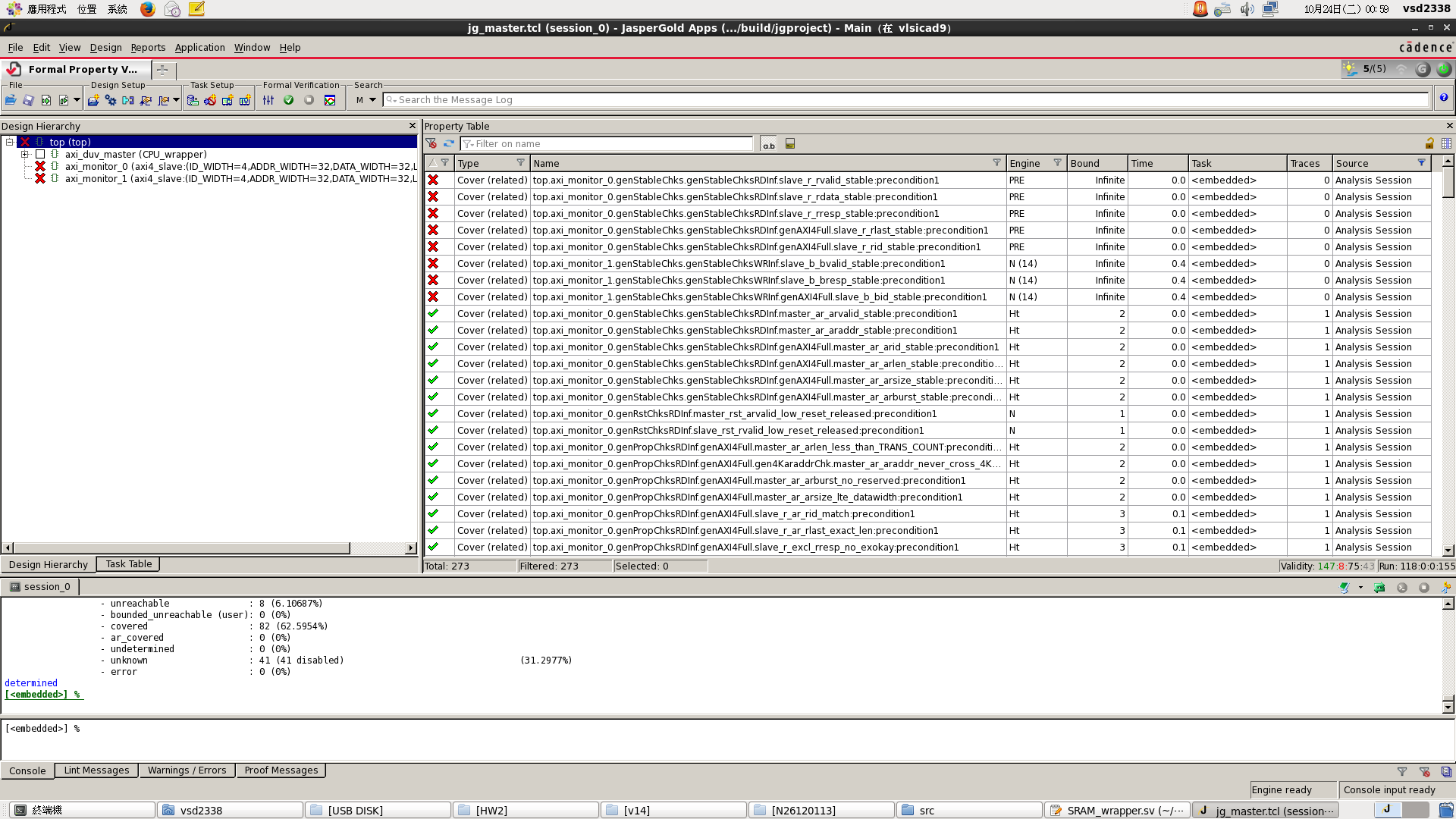
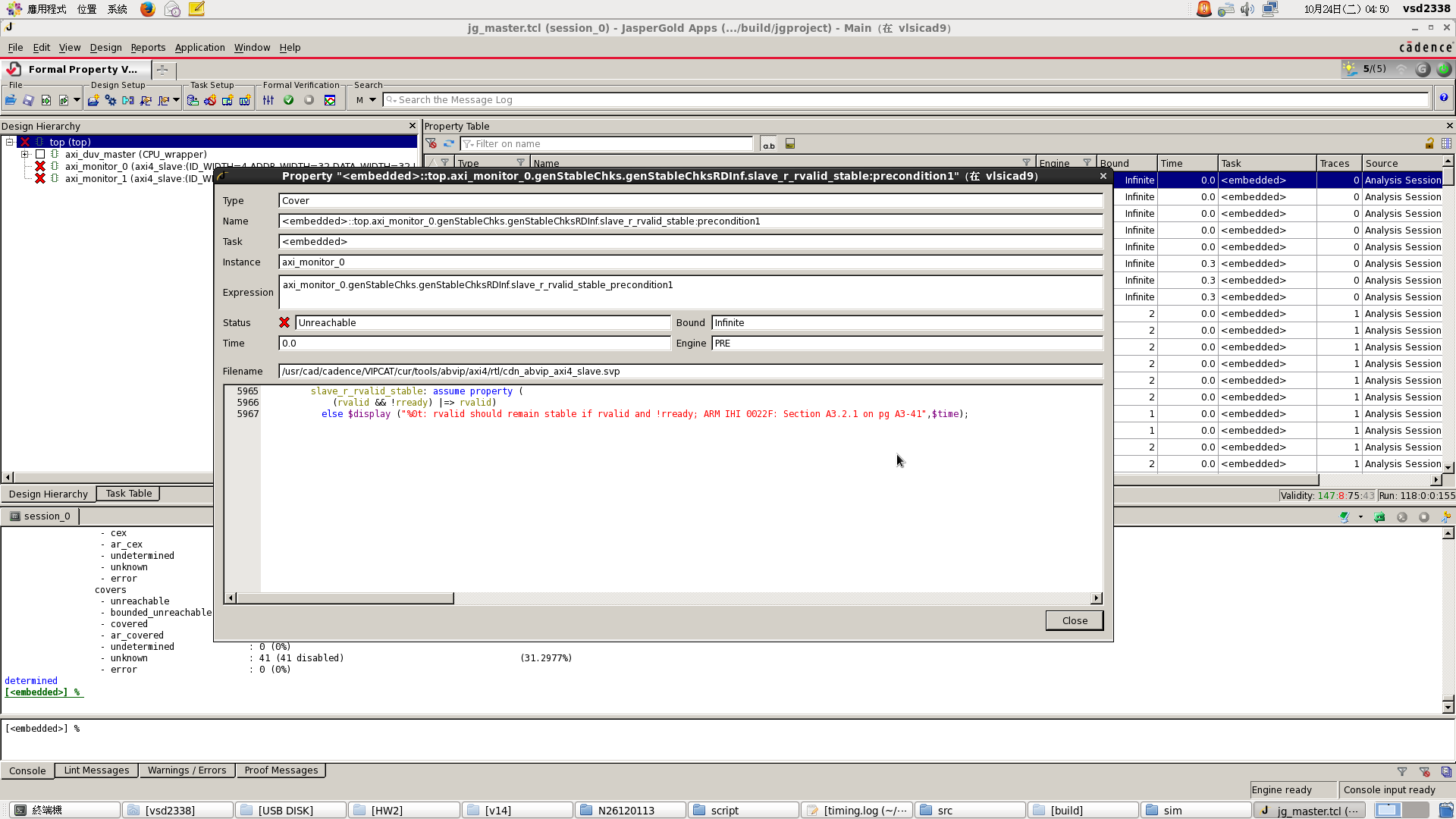
* **CPU wrapper**

Assertion all proven，但有8個前提不成立導致沒cover到。

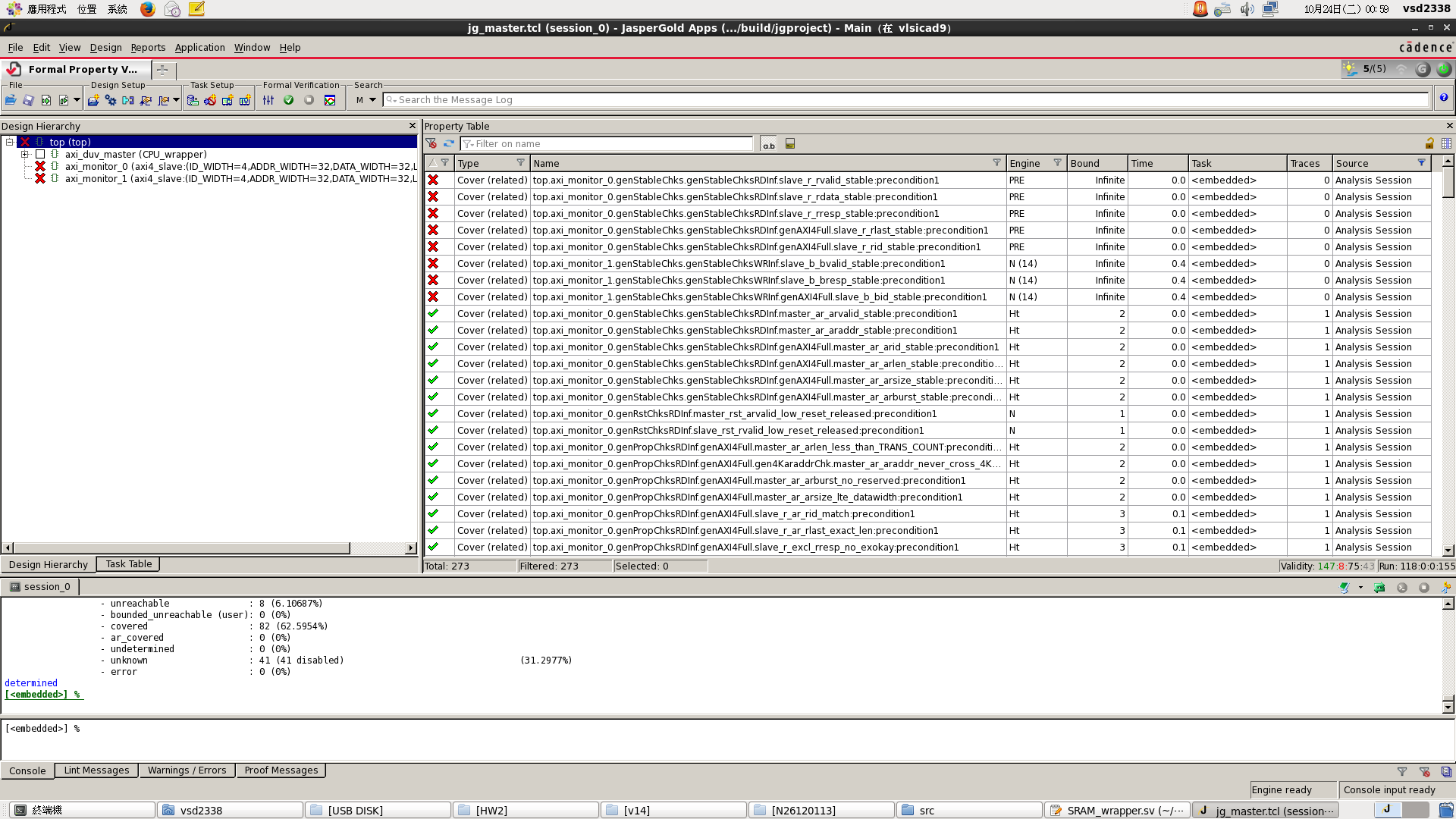
****

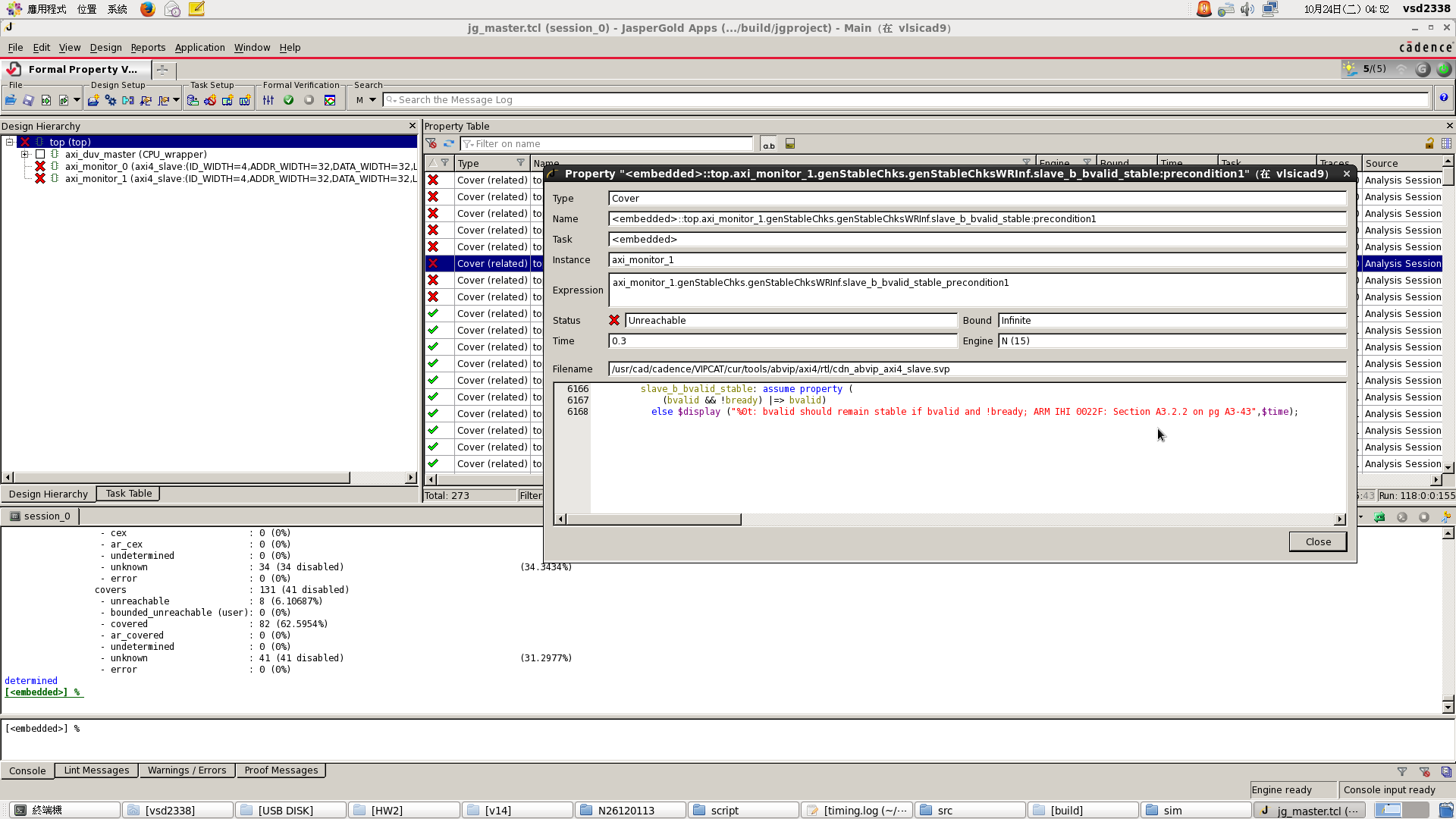
前提不會發生的原因:

(1~5)因為Master0的RREADY一直維持high，故不會發生RRVALID & !RREADY的狀況。

****

(6~8)因為Master1的BREADY在W/B phase時為high，故當BVALID在B phase拉起時，BREADY也是high，不會發生BVALID & ! BREADY的狀況。

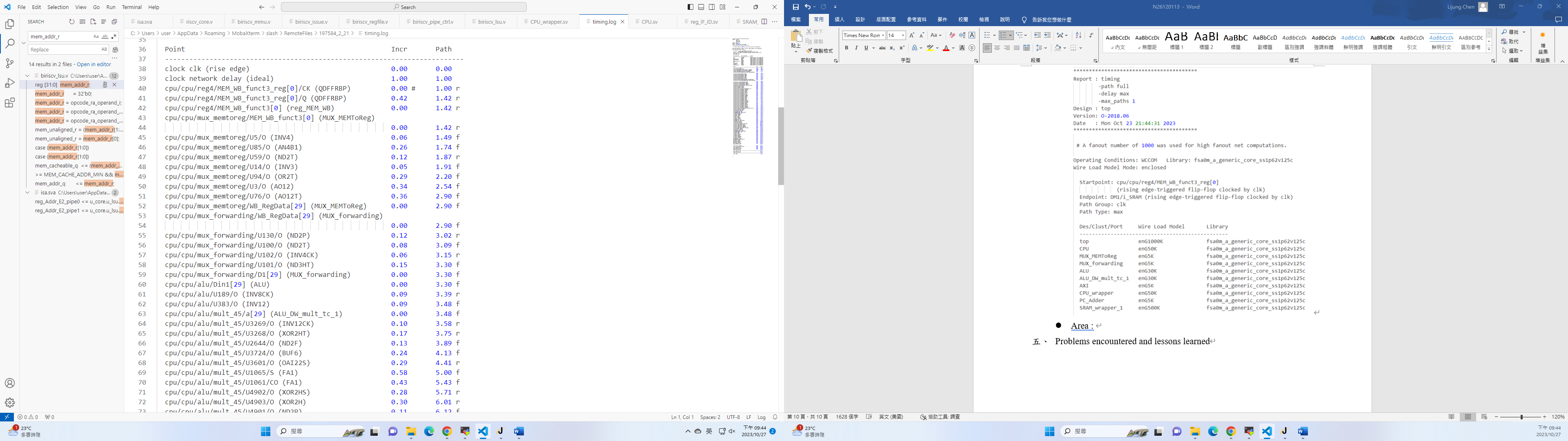
****

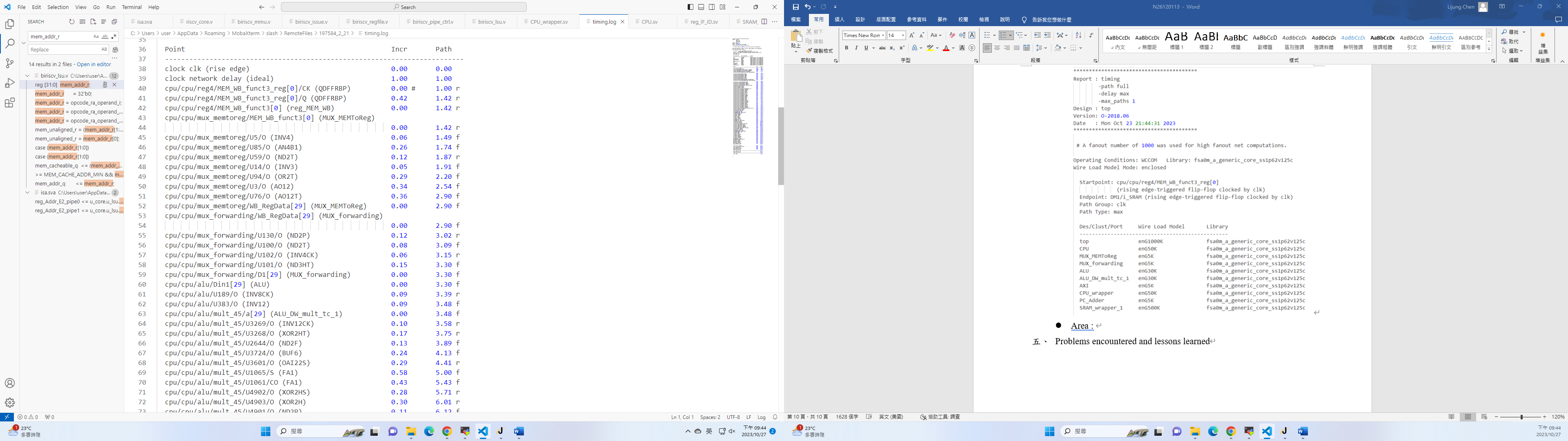


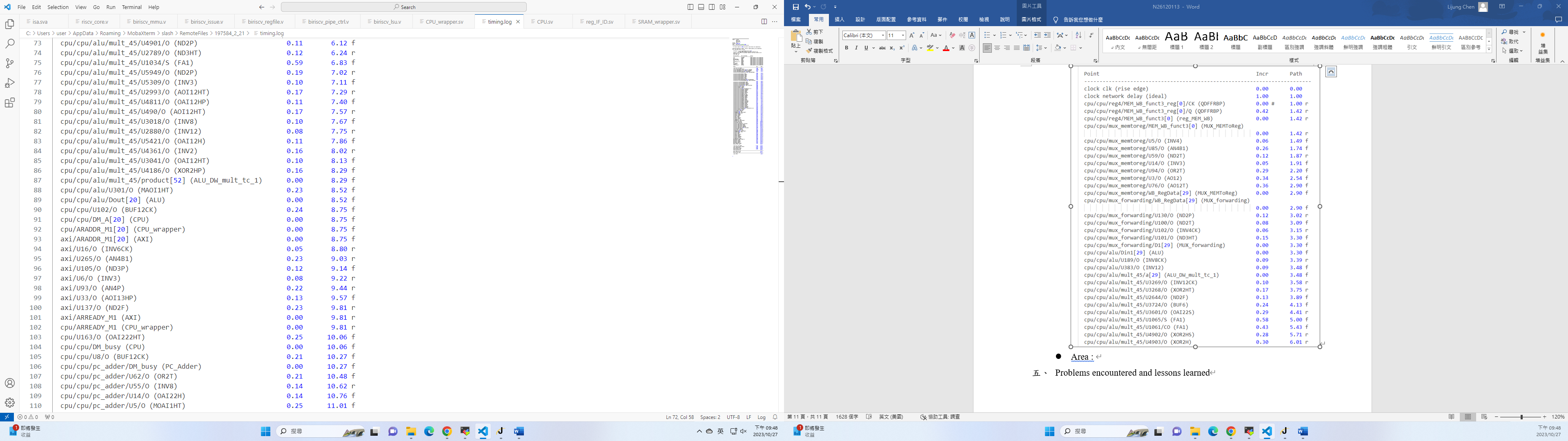
1. Synthesize result

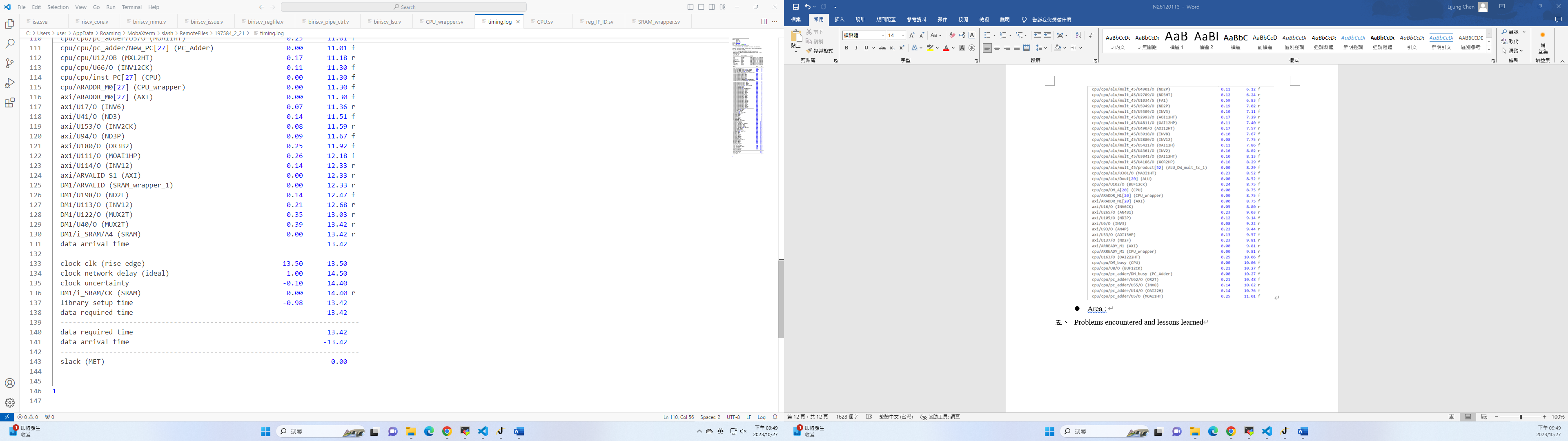
* Cycle time : 13.5 ns





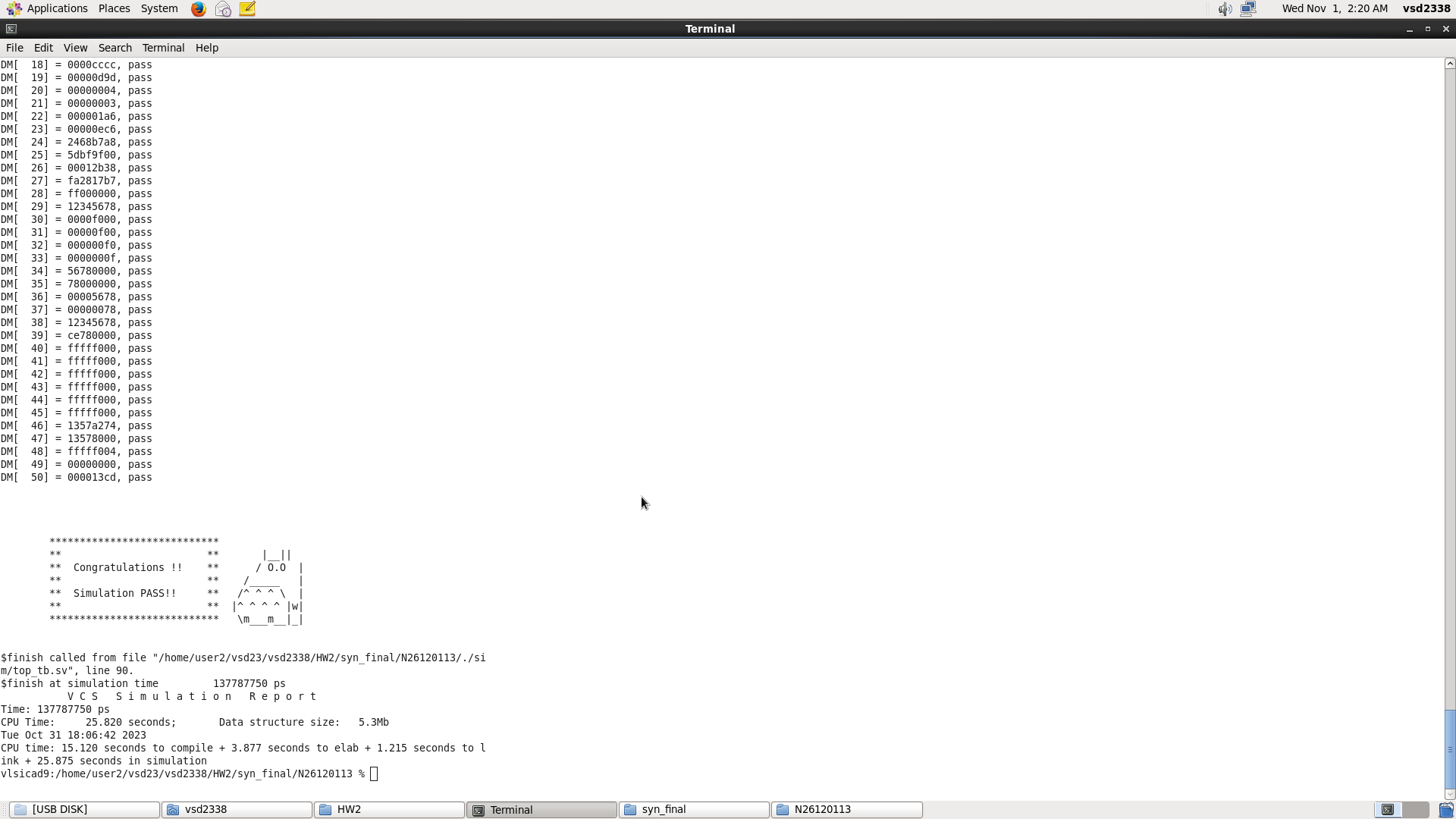




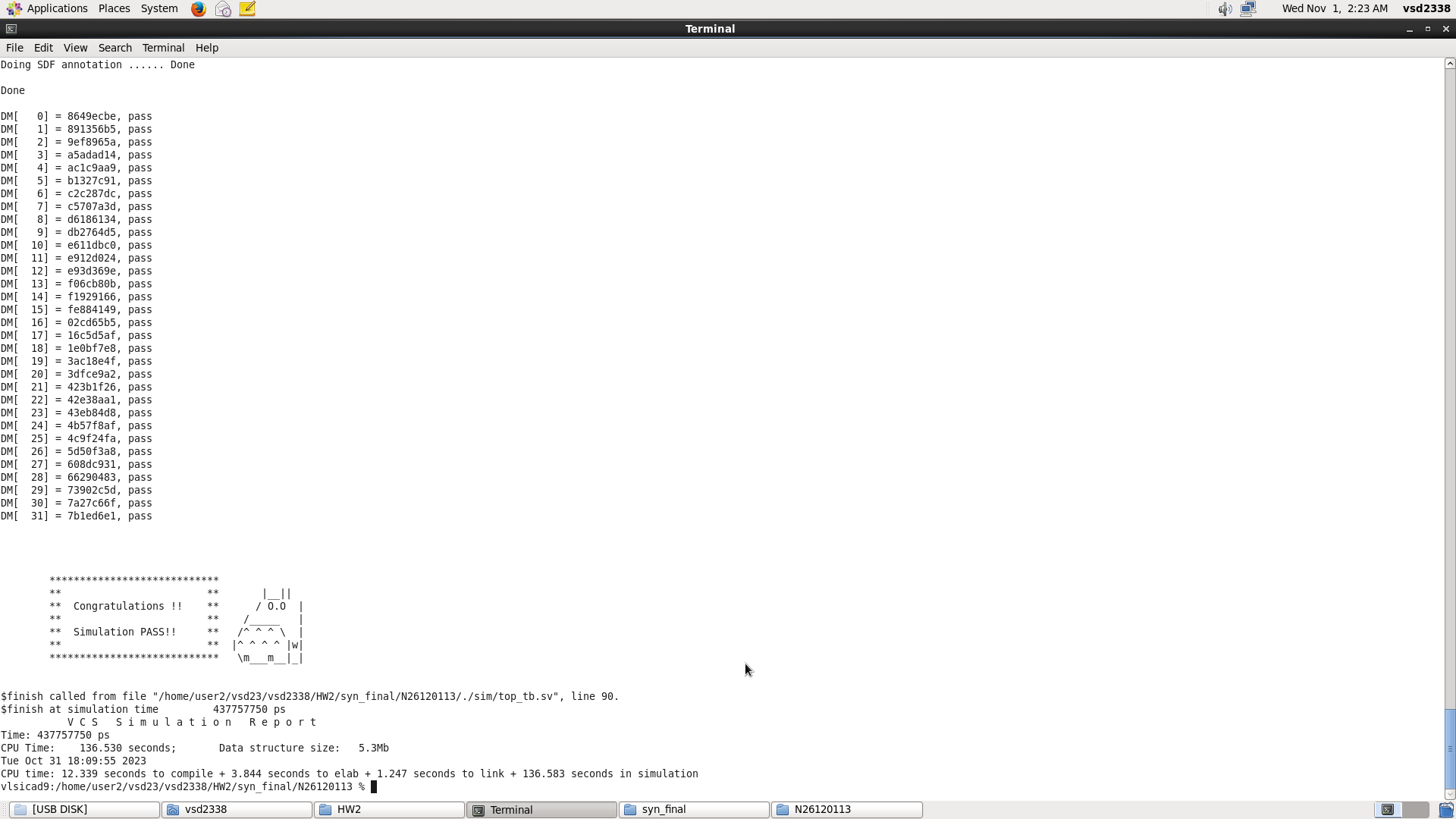


* Simulation time

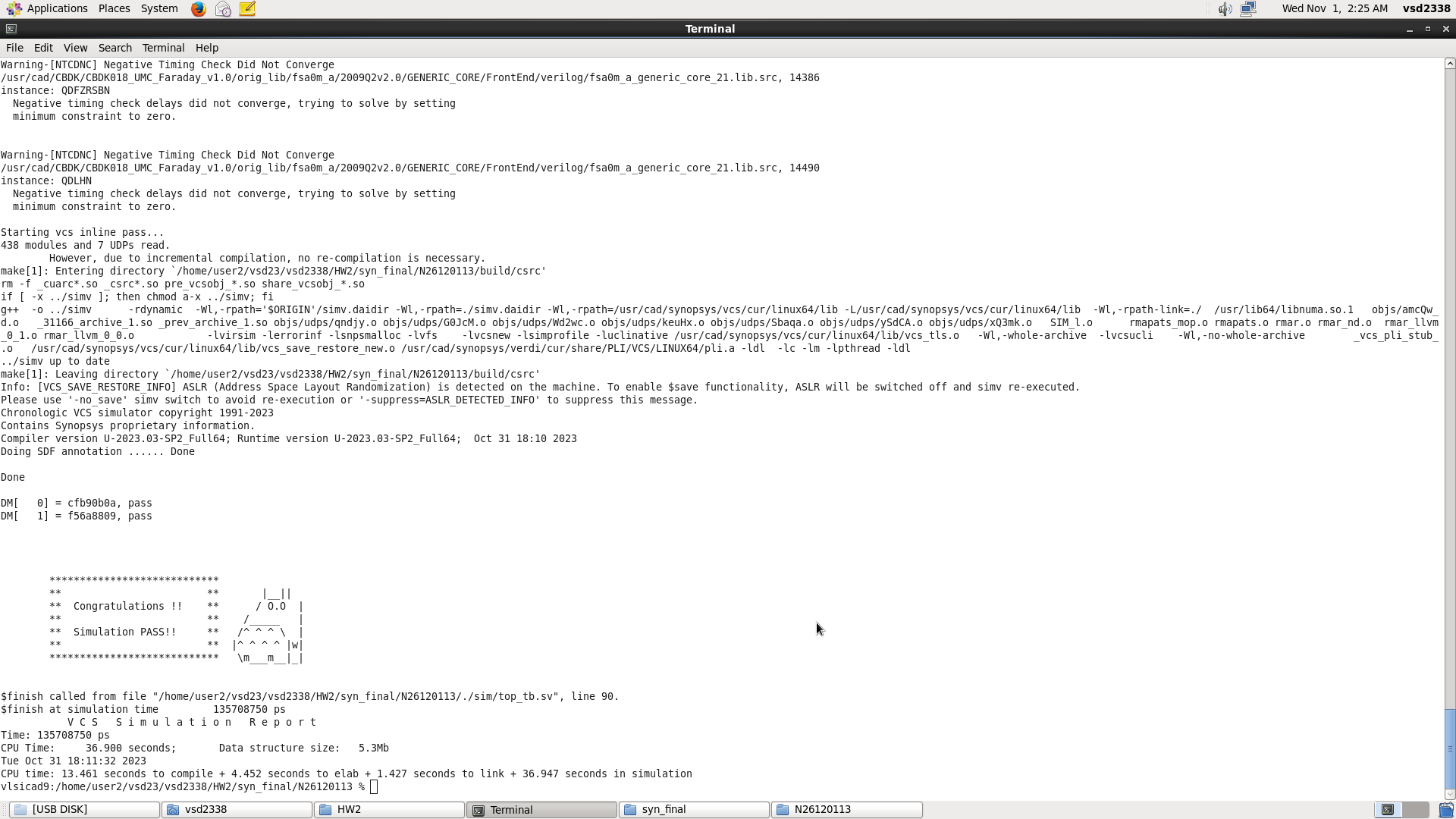
Prog0 : 137787750 ps



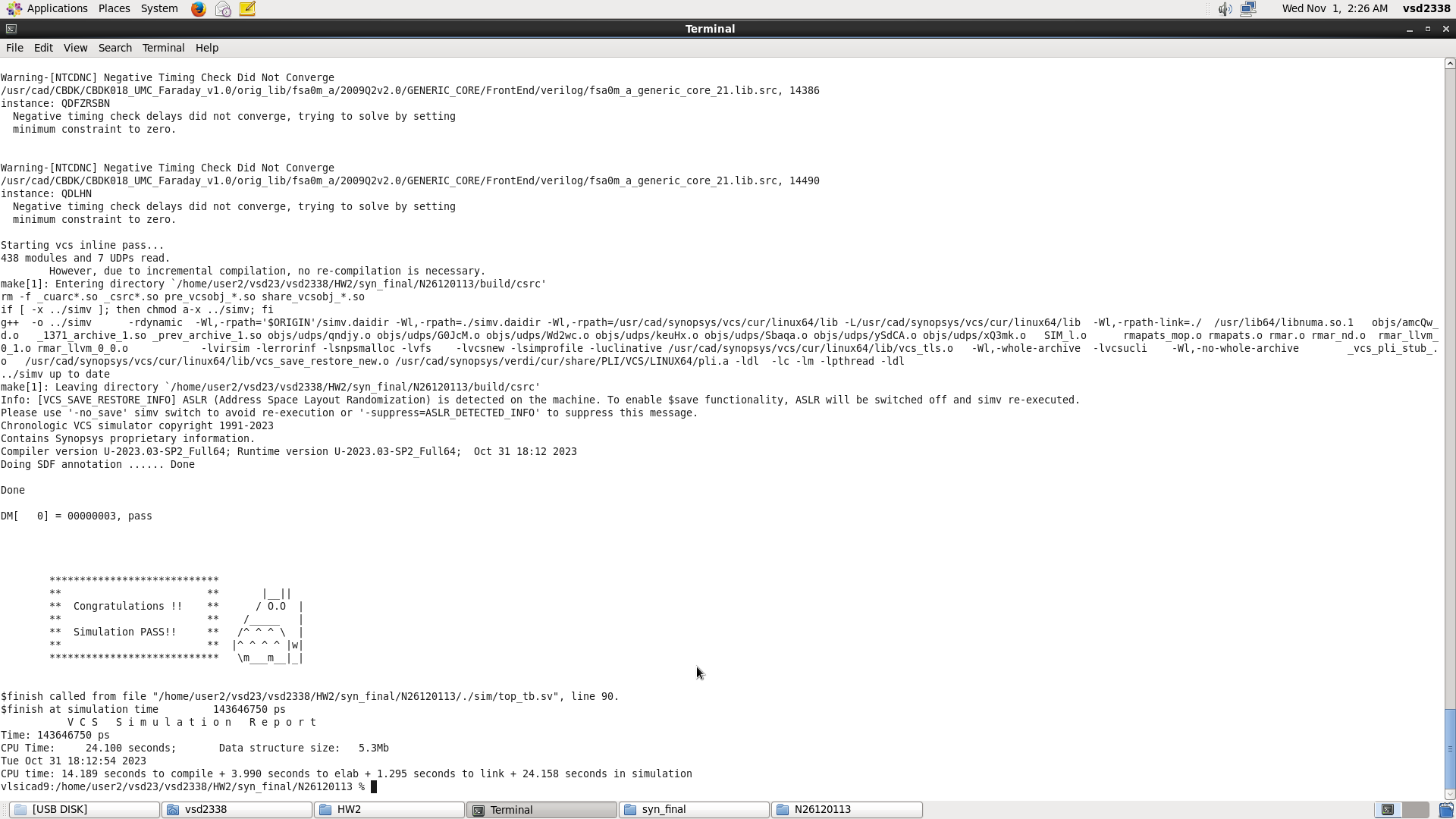
Prog1 : 437757750 ps



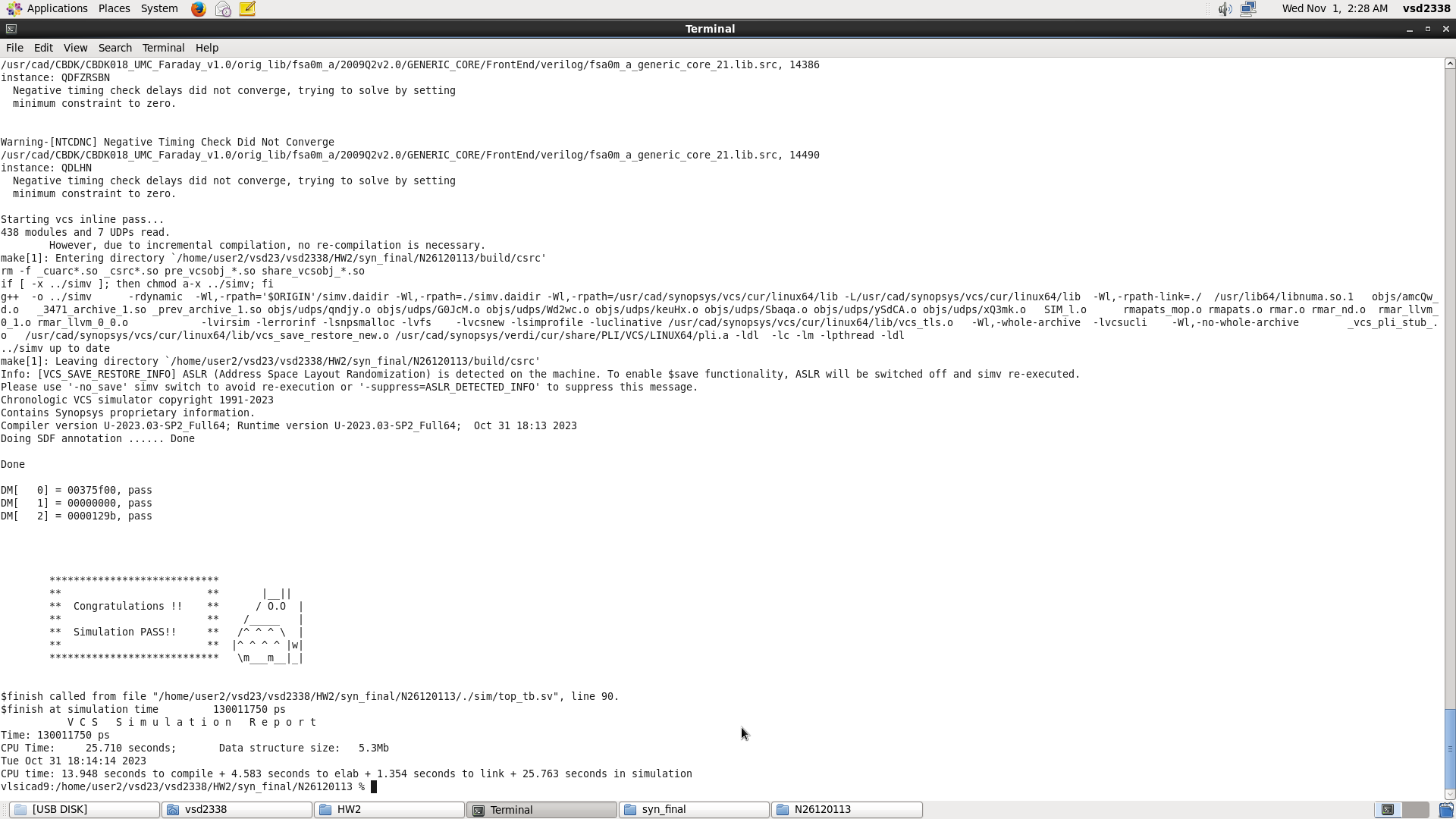
Prog2 : 135708750 ps



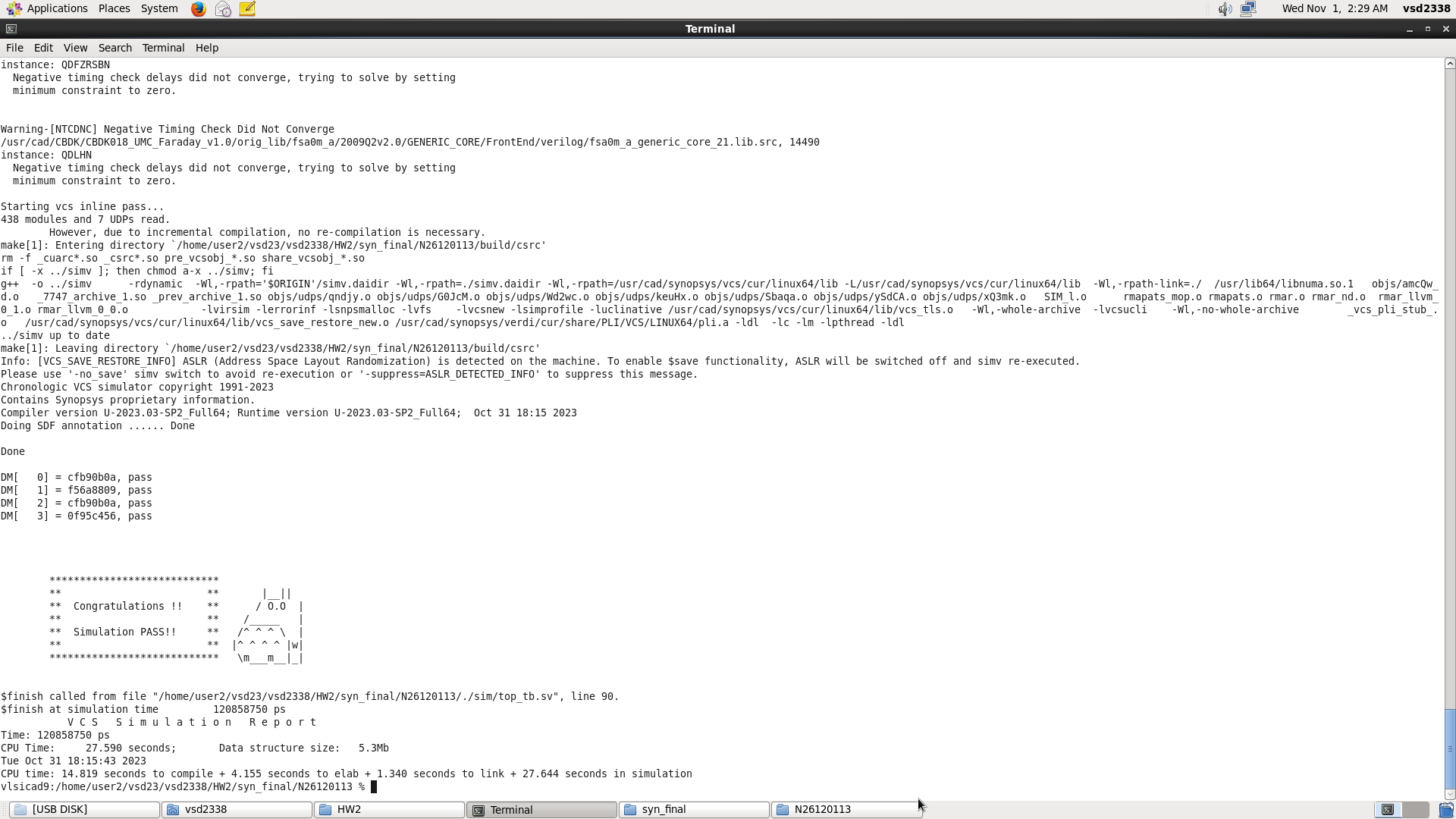
Prog3 : 143646750 ps



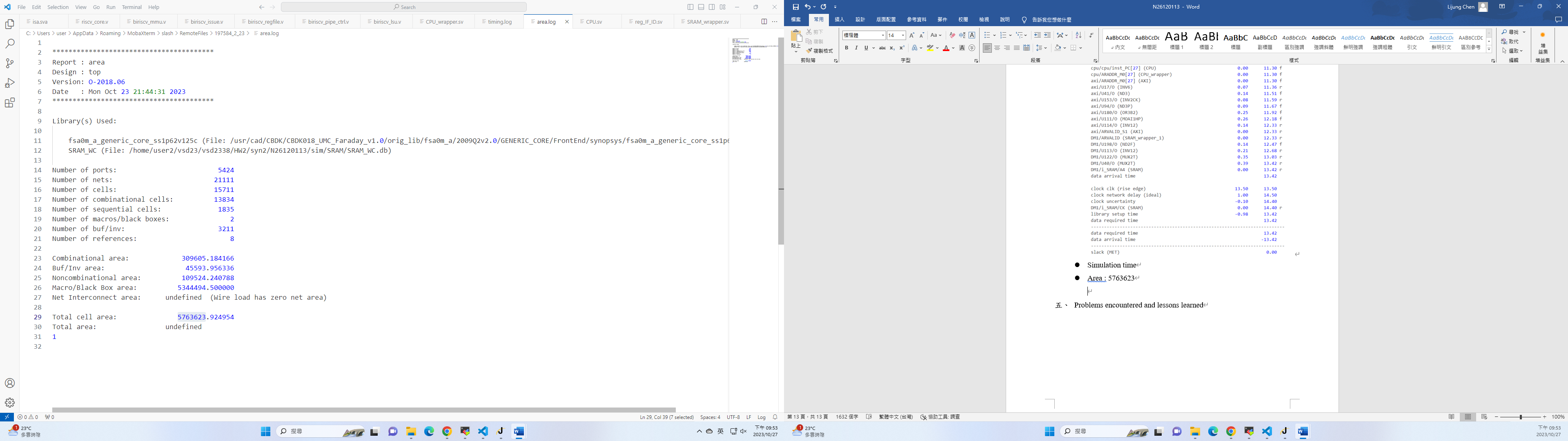
Prog4 : 130011750 ps



Prog5 : 120858750 ps



* Area : 5763623



1. Problems encountered and lessons learned
2. 剛開始不太了解加AXI的作用，以為Master0只會讀寫Slave0，Master1只會讀寫Slave1，所以直接把AXI裡的訊號寫死，把Slave0的訊號給Master0，Slave1的訊號給Master1，但其實Master1也是可以讀取Slave0的，甚至可能發生兩個Master想同時讀取同一個Slave，這時才知道AXI的功能是在分配Master的讀寫Slave的優先權，需要用MUX來決定哪些訊號可以上BUS，而且AXI中不包含暫存功能。
3. 在驗VIP的時候發現訊號Stable的assertion最常發生問題，如果是wrapper中有Stable問題，通常是:
4. VALID拉起後在還沒READY前就放下。
5. VALID在不對的時間拉起，channel對應的VALID訊號只能在對應的phase拉起，例如: AWVALID / ARVALID只能在AW / AR phase拉起，不可在W/B /R phase拉起；WVALID / RVALID只能在W/ R phase拉起，不可在AW / AR phase拉起。
6. 傳輸的資料沒有在wrapper暫存。

如果是在AXI中有Stable問題，不是因為訊號沒有在AXI中暫存(AXI不提供暫存功能)，通常是狀態機的狀態轉換有誤，例如:W phase其實還沒結束，但AXI的狀態機已提前轉換為AW phase，如果原本在W phase時Slave0是要接收Master1的訊號，但因為此時突然跳到AW phase，可能會接收成其他也想使用Slave0的Master的訊號，這樣會導致訊號分配錯誤。

1. CPU wrapper我有做了兩個版本，一個是AWVALID / ARVALID都可以在EX stage拉起，address可以在EX stage就送出，另一個是AWVALID / ARVALID在MEM stage才拉起，前者訊號因為直接從ALU 接出來，導致路徑很長，合成cycle time只能到13.5 ns，後者因為訊號是從MEM pipeline出來，路徑比較短，合成cycle time可以到11.5 ns，但因為前者在讀寫時比後者少一個cycle，所以雖然cycle time比較長，但總cycle數比較少，所以如果讀寫指令變多的話，前者的執行時間會比較快，prog0到prog5的執行時間也幾乎是前者較快。