VLSI System Design (Graduate Level)

Fall 2018

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: \_\_\_\_\_杜冠勳\_\_\_\_\_\_\_\_\_\_\_

Student ID: \_\_\_\_\_N26094883 \_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| Cache Architecture | |
| FSM(READ) | |
| Figure 1 | |
| STATE EXPLANATION | |
| IDLE | Waiting state ,if there is any cpu read signal ,the next state will be check hit read |
| CHECK\_HIT\_READ | Check if the search in cache is hit or not  the hit condition will be (TA\_out==TA\_in )&&(single\_valid\_data)  if miss the next condition will be send read mem1  if hit the next condition will be IDLE |
| SEND\_READ\_MEM1~4 | The state will send the request to cpu wrapper to get data from main memory and the next state will be read\_mem |
| READ\_MEM1~4 | The state is waiting for the finsh of reading data from main memory after reading a cache line data(128bits) reading 4 times from main memory the next state will return to IDLE state |
| Detailed | Because when the design of CPU\_wrapper receive the request from cache ,it will active D\_wait at the same time thus it chould have two state ,one for sending the request signal  One for waiting the D\_wait signal to drop.  More detailed is explan by waveform |
| Special design | Whether the cpu hit or not the cache read the data array by index  however only hit condition will return the data from data array.  In normal case the controller should know the hit condition first and send the read signal to dataarray  In my case ,sending read signal to tag array for checking hit condition and getting data from data array are happen at the same time. |

|  |  |
| --- | --- |
| Cache Architecture | |
| FSM(READ) | |
| Figure 2 | |
| STATE EXPLANATION | |
| IDLE | Waiting state ,if there is any cpu read signal ,the next state will be check hit write |
| CHECK\_HIT\_WRITE | Check if the search in cache is hit or not .  the hit condition will be (TA\_out==TA\_in )&&(single\_valid\_data).  if hit the next condition will be WRITE\_HIT and send the wrinting request to CPU\_wrapper rat the same time.  if hit the next condition will be WRITE\_MISS and send the wrinting request to CPU\_wrapperat the same time. |
| WRITE\_HIT | Write through to the memory and cache  Check if writing operation to main memory is finish or not .  If finish it will return to IDLE |
| WRITE\_MISS | Write through to the memory  Check if writing operation to main memory is finish or not .  If finish it will return to IDLE |
| Detailed | Because when the design of CPU\_wrapper receive the request from cache ,it will active D\_wait at the same time thus it chould have two state ,one for sending the request signal  One for waiting the D\_wait signal to drop.  More detailed is explan by waveform |

|  |  |
| --- | --- |
| Waveform | |
| Cache | |
| Figure 3 | |
| Explanation | |
| When the cpu send req to cache the core wait active(green block) | |
| READ\_miss | (red block)  IM read instruction  When the cpu start the cache has no data thus write miss happen  When the write miss happen ,cache send request to cpu wrapper it can explan by wrapper ARADDR get the address sent by cpu 0 4 8 c  The wrapper send the data to cache  if(4'd0<=offset&&offset<4'd4) it will return the first data to cpu that get from CPU\_wrapper which is this case  if(4'd4<=offset&&offset<4'd8) it will return the second data that get from CPU wrapper to cpu  if(4'd8<=offset&&offset<4'd12) it will return the third data that get from CPU wrapper to cpu  else it will return the forth data that get from CPU wrapper to cpu |
| READ\_hit | (yellow block)  Because there is a write miss at reading address=0 so  The cache will update one cache line which address=0 4 8 C strored in cache  Therefore when reading address = 0 4 8 hit operation should be hit in cache which can proved by the core wait time is obviously shorten than privous. |

|  |  |
| --- | --- |
| Waveform | |
| Cache | |
| Figure 4 | |
| Explanation | |
| When the cpu send req to cache the core wait active(green block) | |
| WRITE\_miss  (Figure 5) | (red block)  74C is the store instruction :SW the third stage counted by cpu\_stall sending the write requset to cache controller  When the cpu send the write signal ,the FSM state:STATE\_CHECK\_HIT\_WRITE(8) observe that if the hit hit condition happened or not  When the write miss happen:state(9),the controller will not send the write signal to data array, valid array and tag array (PINK BLOCK)  but it send the write signal to CPU WRAPPER (YELLOW BLOCK)  The State will wait the D\_wait signal sent by CPU wrapper back to zero and return to IDLE STATE(0) (red block)  The cpu\_wait signal is still active high because instruction cache is reading data from memory |

|  |  |
| --- | --- |
| Waveform | |
| Cache | |
| Figure 6 | |
| Explanation | |
| When the cpu send req to cache the core wait active(green block) | |
| WRITE\_hit | (red block)  7fc is the store instruction :SW the third stage counted by cpu\_stall sending the write requset to cache controller  When the cpu send the write signal ,the FSM state:STATE\_CHECK\_HIT\_WRITE(8) observe that if the hit hit condition happened or not  When the write hit happen:state(10),the controller will send the write signal to data array(PINK BLOCK)  The reson why web data is f3ff is because it’s SH instruction  Web data is also contrill by offset data  The write hot operation also send the write signal to CPU WRAPPER (YELLOW BLOCK)  The State will wait the D\_wait signal sent by CPU wrapper back to zero and return to IDLE STATE(0) (red block)  The cpu\_wait signal is still active high because instruction cache is reading data from memory |

|  |  |  |
| --- | --- | --- |
| Waveform | | |
| Instruction | | LH |
| ff041383 | | lh t2,-16(s0) |
| Figure 7    Figure 8 | | |
| Explanation | | |
| ID | The ID stage decode opcode and decode read mem signal  The mem\_out\_lowis active and the mem\_out\_half ,padding zero is not active because it ‘s the load half word(signed extended) instruction(red block) | |
| EXE | The EXE stage calculate the address that is going to read from memory(green block) | |
| MEM | READ data from cache because the data is hit in cache(blue block)  Read data is core\_out=ccccccc | |
| WB | Because the lh operation and the address[1]=0 thus the write back data shold be ffffcccc(yellow part) | |

|  |  |  |
| --- | --- | --- |
| Waveform | | |
| Instruction | | LBU |
| ff044e03 | | lbu t3,-16(s0) |
| Figure 9    Figure 10 | | |
| Explanation | | |
| ID | The ID stage decode opcode and decode read mem signal  The mem\_out\_half padding zero is active and the mem\_out\_low is not active because it ‘s the load byte word (unsigned signed extended)instruction(red block) | |
| EXE | The EXE stage calculate the address that is going to read from memory(green block) | |
| MEM | READ data from cache because the data is hit in cache(blue block)Read data is core\_out=ccccccc | |
| WB | Because the lh operation and the address[1:0]=00 thus the write back data shold be 000000cc(yellow part) | |

|  |  |  |
| --- | --- | --- |
| Waveform | | |
| Instruction | | LHU |
| ffe41723 sh t5,-18(s0) | | lhu t4,-16(s0) |
| Figure 11    Figure 12 | | |
| Explanation | | |
| ID | The ID stage decode opcode and decode read mem signal  The mem\_out\_half padding zero is active and the mem\_out\_low is not active because it ‘s the load half word (unsigned signed extended)instruction(red block) | |
| EXE | The EXE stage calculate the address that is going to read from memory(green block) | |
| MEM | READ data from cache because the data is hit in cache(blue block)Read data is core\_out=ccccccc | |
| WB | Because the lh operation and the address[1]=0 thus the write back data shold be 0000cccc(yellow part) | |

|  |  |  |
| --- | --- | --- |
| Waveform | | |
| Instruction | | SH |
| ffe41723 | | sh t5,-18(s0) |
| Figure 13    Figure 14 | | |
| Explanation | | |
| ID | The ID stage decode opcode and decode read mem signal  The mem\_in\_half is active (yellow block) | |
| EXE | NONE | |
| MEM | CPU send the write signal to cache to write data to memory .  In this case the cache hit condition happened which can proved by the FSM state 0 >8>a(blue block)  Data array.WEB is f3ff is controlled by address[1] and core type=5 sent by CPU (green block)  MEMORY.WEB if 3 is controlled by address[1] and core type=5 sent by cache (red block)  Web data :4’h3=4’b0011 which can proved SH store half word | |
| WB | NONE | |

|  |  |
| --- | --- |
| Simulation(under20ns) | |
| **rtl0** | **rtl1** |
| **rtl2** | **Syn0** |
| **Syn1** | **Syn2** |

|  |  |
| --- | --- |
| Simulation | |
| **pr0** | Pr1 still runing |
| Pr2 still runing |  |

|  |
| --- |
| JASPERGOLD |
|  |
| Explanation |
| In my case when the core send write req to cache it will send send D\_req for only one cycle .  The reson why I only keep D\_req for only one cycle is due to my CPU wrapper design.  If the read signal(D\_req) is always is keep active during write or read operation ,the CPU wrapper will lock in cycle for reading or writing .  The lock condition is due to my CPU\_wrapper 0 STATE.In 0 STATE The CPU\_wrapper receive request will active D\_wait at the same time because it’s combinaional circuit.  Therefore in my cache design ,I have to keep my request for only 1 cycle preventing from recursive condition.  Nevertheless ,the jaspergold verification has the assume that D\_req will remain high during D\_wait which will not cover my design. |

|  |  |
| --- | --- |
| Synthesize result(8ns) | |
| TIMING |  |

|  |  |
| --- | --- |
| AREA |  |
| POWER |  |

|  |  |
| --- | --- |
| APR | |
| Before Placement  Verify Geometric |  |
| Before Placement  Verify Connectivity |
| Post\_route  Setup\_verify |  |
| Post\_route  holdup\_verify |  |

|  |  |
| --- | --- |
| Post\_cts  Setup |  |
| Post\_cts  Holdup |  |
| ANTENNA  &  GEOMETRIC  CONNECTIVITY  &  VERIFY |  |
| violation | 2VDD is too close to VDD  2GND is too close to GND  The violation won’t affect the verification |

|  |
| --- |
| Floorplan\_View |
|  |
| Amoeba\_View |
|  |
| Physical\_view |
|  |

|  |  |
| --- | --- |
| Superlint | |
|  | |
| Warning Explantion | File format 為命名沒有和module 同名 不影響合成  Coding style 是因為寫法習慣的問題 combination 電路中 會有接線來自 sequential 的區塊 不影響合成 一樣是合出flipflop |
| Superlint within 95% | 1-(49+23)/9923 = 99.27% >95% |

|  |
| --- |
| 心得 |
| 我從這次作業中學習到cache 的設計,其中有諸多考量是當時學習計組時不會想到的,像是 在更新cache line 時 因為 128 bits 在 hits 要回傳資料時 不該用 offset 為 0 4 8 c 的方式判斷要回傳哪筆資料 而是應該用0~3 4~7 8~b c~f 的 方式去做判斷 因為 address 有可能不是4的倍數 像是 LB SB SH LH 都有可能計算出來的address 不是4 的倍數 另外這次作業中因為讀取SRAM DATA 時 回傳資料為32 bits 導致 READ MISS 的時候 要讀取 MEMORY 四次 才能回傳資料 增加了許多時間成本 而作業四聽老師上課說道 我們可以控制回傳的bandwidth 不但可以讓MEMORY 更為省電 也會減少MISS 所產生的time panalty 另一項設計讓time panalty 得以下降 的設計是 我不論read hit 或是 read miss 都會在判斷的時候同時讀取 data memory 的值 而不會等到判斷確定是read hit 才向data memory 進行讀取而多耗費一個cycle,進一步達到減少time panalty 的效果  經由這次作業讓我對整個CPU 系統架構有更進一步的認識。 |