

Cell-Based Training LAB

<Verilog> LAB

Nov. 2012 Y. X. Chen



Cell-Based Training LAB

LAB File List					
Folder	Name	Description			
lab01	RAM_64B.v	Memory verilog model			
	cpu_bug.v	CPU RTL design with bugs			
	tcpu.v	Testbench			
	cpu.v	CPU RTL design			
lab01/example	counter.v	Counter RTL design			
	counter_bug.v				
	counter_tb3.v				

Before your Lab:

unix%> mkdir LAB01 unix%> <u>cd LAB01</u> unix%> <u>cp /home/areslab/LAB_12/lab01.tar.gz .</u> unix%> <u>tar -xvf lab01.tar.gz</u>

Example:

Ex 1-1

- unix% > cd lab01/example
- unix%> <u>nLint -gui</u>
- Select $\langle File \rangle \rightarrow$ Import Design...
 - <From File> Tag
 - Select cpu_bug.v
 - OK
- Select <Run $> \rightarrow$ Lint





<u>Ex 1-2</u>

- Run RTL simulation
 - unix%> $\underline{ncverilog} + \underline{access} + r counter_tb.v$

If any error occurs, please check your testbench and your design.

الم vlsi01@linuxserver: ~			×		
linuxserver:~/lab01/lab01/example> ncverilog +access+r counter_tb3.v					
ncverilog: 10.20-s114: (c) Copyright 1995-2012 Cadence Design Systems,	, Inc				
Loading snapshot worklib.counter_tb:v					
<pre>ncsim> source /usr/cad/cadence/INCISIV/cur/tools/inca/files/ncsimrc</pre>					
ncsim> run					
Novas FSDB Dumper for Verilog-XL, Release 2008.01 (Linux) 01/30/2008					
Copyright (C) 1996 - 2007 by Novas Software, Inc.					
Novas Create FSDB file 'counter.fsdb'					
Novas Begin traversing the top scope(counter_tb), layer(0).					
Novas End of traversing the top scope(counter_tb)					
Simulation complete via \$finish(1) at time 100 NS + 0					
./counter_tb3.v:39 #100 \$finish;					
ncsim> exit					

- Waveform viewer
 - unix%><u>nWave &</u>
 - Open file: $\langle File \rangle \rightarrow Open File \rightarrow Choose$ "tcpu.fsdb"
 - Select Signals: <Signal>→Get Signal
 - Choose desired module and signals
 - Save waveform: $\langle File \rangle \rightarrow save signal$
 - In the top of blank, type "filename.rc" in the end of path
 - OK





Cell-Based Training LAB

<u>Lab 01</u> <u>Specification</u>

1. Block diagram



2. Instruction set

	Instruction Code	Meaning
C=A+B	0000	將暫存器A與B中的數值相加放進暫存器C
C=A-B	0001	將暫存器A與B中的數值相減放進暫存器C
C=A+1	0010	將暫存器 A 中的數值加 1 放進暫存器 C
C=A-I	0011	將暫存器 A 中的數值減 1 放進暫存器 C
C=A+B+I	0100	暫存器 A、B 數值相加再加 1 放進暫存器 C
C = -A	0101	暫存器C的值辦換為暫存器A值的負數
Input A	0110	輸入的資料放進暫存器A
Input B	0111	輸入的資料放進暫存器 B
Input Memory	1000	輸入的資料放進記憶體中
Storage C	1001	將暫存器C中的資料存入記憶體中
Load C	1010	將記憶體中的資料存入暫存器 C 中
Output C	1011	將暫存器C中的資料輸出
Output Mem	1100	將記憶體中資料輸出
A=C	1101	將暫存器C中的值放進暫存器A中
B=C	1110	將暫存器C中的值放進暫存器B中
customize	1111	設計者自定功能

3. Memory operation

a. Read operation





b. Read operation



CEN	WEN	OEN	Data Out	Mode	Function
н	х	L	Last Data	Standby	Address inputs are disabled; data stored in the memory is retained, but the memory cannot be accessed for new reads or writes. Data outputs remain stable.
L	L	L	Data In	Write	Data on the data input bus D[n-1:0] is written to the memory location specified on the address bus A[m-1:0], and driven through to the data output bus Q[n-1:0].
L	н	L	SRAM Data	Read	Data on the data output bus Q[n-1:0] is read from the memory location specified on the address bus A[m-1:0].
х	х	н	Z	High-Z	The data output bus Q[n-1:0] is placed in a high impedance state. Other memory operations are unaffected.

<u>Lab 1-1</u>

- 1. Unix%> <u>nLint –gui &</u>
- 2. Select $\langle File \rangle \rightarrow$ Import Design...

a. <u><From File> Tag</u>

- b. Select cpu_bug.v
- c. OK
- 3. Select <Run $> \rightarrow$ Lint
- 4. The Lint report is shown

Lab 1-2

- 1. Fix the bugs of cpu_bug.v
- 2. Write the testbench in tcpu.v
- 3. Run RTL simulation
 - unix% > ncverilog + access + r tcpu.v

If any error occurs, please check your testbench and your design.

- 4. unix%> <u>nWave &</u>
- 5. Open file: $\langle File \rangle \rightarrow Open File \rightarrow Choose "tcpu.fsdb"$



- 6. Select Signals: <Signal>→Get Signal Choose desired module and signals
- 7. Save waveform: $\langle File \rangle \rightarrow save signal$
 - a. In the top of blank, type "filename.rc" in the end of path
 - b. OK

Lab l Requirements:

- 1. cpu_bug.v 之 nLint 結果, 說明錯誤或警告的原因, 並探討如何修正
- R據 cpu_bug.v 之 nLint 結果,將 code 更正為 cpu.v,並利用 nLint tool 測試 cpu.v 之 verilog code,所得到的結果必須是 0 error (若無法達到 0 error,請說 明理由),且功能必須符合 Specification
- 3. 完成 tcpu.v 未寫完的部分,每個指令皆要測試其功能
- 4. RTL 波型
- 5. Deadline: 12/5 上課前繳交