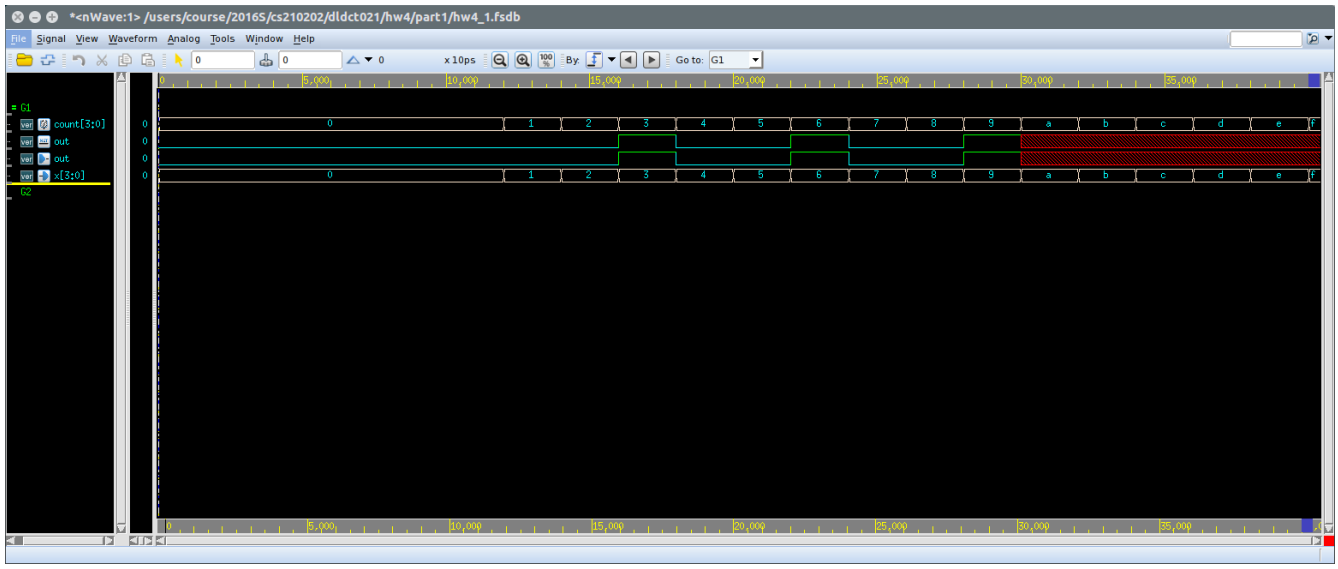
1.

這次的輸出是改用新學到的 monitor 語法，我覺得這樣可以使 testbench 寫起來更簡潔。而且測試的方法是用 repeat 去跑，就不需要手動輸入很多值了，程式碼也更方便閱讀。此外，我覺得程式碼加註解是好習慣，所以關鍵的地方我都有附上註解。下圖是模擬結果，前面我學老師講義上寫#100，因為講義上說為了「wait for global reset if any」，我覺得很有道理就加了。



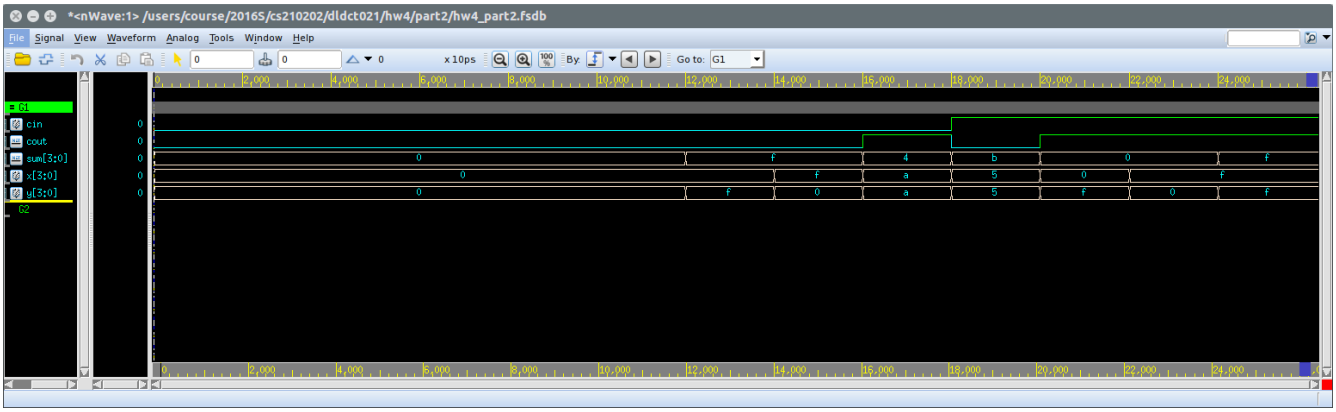
2.

首先是 f_adder module，我覺得 Verilog 語法很新潮，可以很方便就描述 single bit 加法了。而 hw4_2 module，我的作法是把四個 single-bit adder 串起來，就可以做出 four-bit 的加法了。只是串接時，我發現其實接法上非常相似，於是我去請教老師有沒有可能一次生成一組四個，我也因此學到一些額外的黑魔法。

最重要的 hw_2_test module，我只選了 8 個測試。基於 four-bit 加法的概念是四個 single-bit 加法的合成，因此我只需要測每個合成後的 single-bit 是否能正常運作。每個 single-bit 需要 8 個測試，因此理論上需要$8 \times 4 = 32$個測試，但是如果只對每個 single-bit 做測試，其他 bit 就沒有完全使用到，因此這 8 個可以同時測，就可以知道這個 four-bit adder 是否正常運作了。

adder0				adder1				adder2				adder3			
cin	x[0]	y[0]	sum[0]	c1	x[1]	y[1]	sum[1]	c2	x[2]	y[2]	sum[2]	c2	x[3]	y[3]	cout
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	1	0	1	0	0	1	0	1	1	0	1	0	0	1
1	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

上表列出了對於每個 adder 都會測到所有可能，而且只需要八種。


*<nWave:1> /users/course/2016S/cs210202/dldct021/hw4/part2/hw4_part2.fsdb
File Signal View Waveform Analog Tools Window Help
Go to: G1
G1
cin
cout
sum[3:0]
x[3:0]
y[3:0]
G2