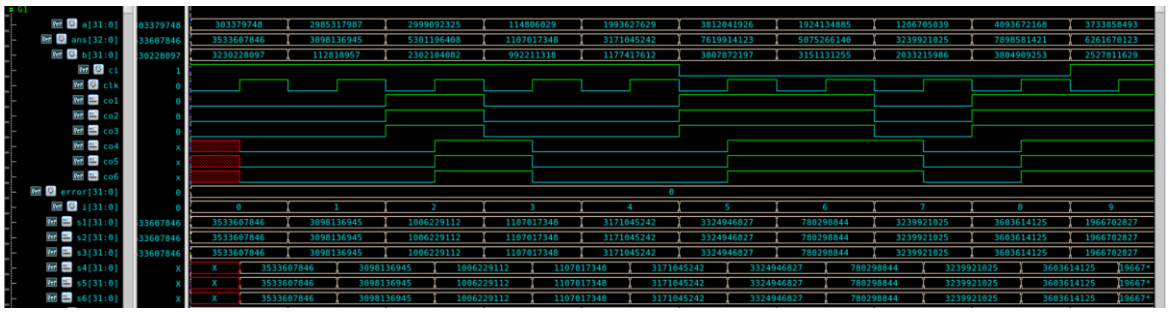
B103040021 謝鎧駿 HW1 Report

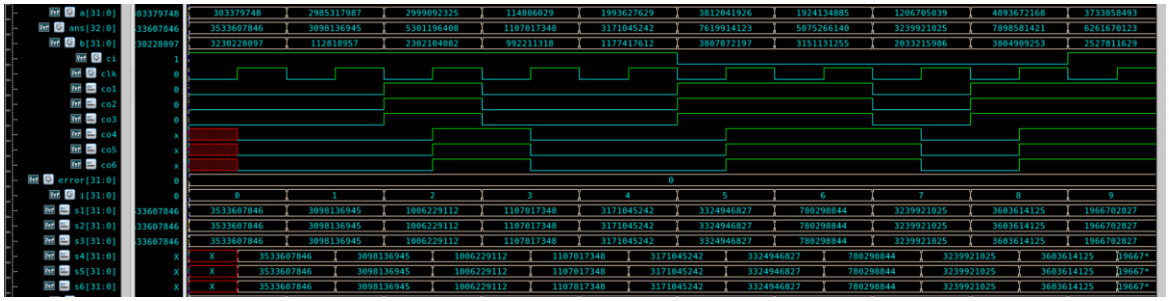
一、設計概念

1. Structure：透過助教在 HW 敘述給的邏輯電路圖去設計 Full adder，再將上課所學的 For 迴圈使用上，變成 32bit 的 adder。
2. Dataflow：這就比較直覺，直接用+的加起來，但要考慮到 carry out。
3. Behavior：用 always 去寫，學到小括號內放*，代表任一 input 改變，就會去計算，而算是跟 dataflow 一樣。
4. DFF：用助教在 HW 敘述給的 RTL code 去設計 1bit 的 DFF，再用 for 迴圈呼叫 1bit 的 DFF 去兜成 32bit 的 DFF。
5. Store in reg：都是先呼叫原本的 adder 後再透過 DFF 去儲存在 register。

二、RTL 波型



三、Gate-level 波型



四、Area/Delay/between 比較數據

		Area (um^2)			Delay (ns)	Power (W)		
		CL	SL	Total		dynamic	leakage	total
adder_structure	delay	92.482562	0	92.482562	0.473287	118.4376	88.0144	206.452
	area	48.107521	0	48.107521	0.76627	36.2086	33.7453	69.9539
	between	67.132802	0	67.132802	0.6197785	59.7882	50.9201	110.7083
adder_structure_reg	delay	88.024322	30.79296	118.817283	0.492727	294.4581	105.3892	399.8473

	area	48.418561	30.792960	79.211521	0.766665	158.4672	56.3295	214.7922
	between	69.258242	30.792960	100.051202	0.629696	208.6694	78.3431	287.0125
adder_dataflow	delay	105.909122	0	105.909122	0.086346	577.5765	95.2921	672.8686
	area	33.488641	0	33.488641	0.890143	23.1684	17.7645	40.9329
	between	52.254721	0	52.254721	0.4882445	53.9102	32.3345	86.2447
adder_dataflow_reg	delay	114.359042	30.792960	145.152002	0.105612	1.3873	133.3770	134.7643
	area	36.028801	30.79296	66.821761	0.885552	129.1030	45.5836	174.6866
	between	51.995521	30.792960	82.788481	0.499076	240.4395	57.239	297.6785
adder_behavior	delay	105.909122	0	105.909122	0.086346	577.5765	95.2921	672.8686
	area	33.488641	0	33.488641	0.890143	23.1684	17.7645	40.9329
	between	52.254721	0	52.254721	0.4882445	53.9102	32.3345	86.2447
adder_behavior_reg	delay	114.359042	30.792960	145.152002	0.105612	1.3873	133.3770	134.7643
	area	36.028801	30.79296	66.821761	0.885552	129.1030	45.5836	174.6866
	between	51.995521	30.792960	82.788481	0.499076	240.4395	57.239	297.6785

五、觀察三種 modeling 的數據與波型是否相同，解釋你認為的原因：

1. 首先，我發現我 adder_dataflow 與 adder_behavior 的數據完全相同，我認為是因為 dataflow 與 behavior 的寫法差不了太多，並且算式都只有一行，所以得出的所有數據都相同。

2. 最小 Area 跟預料的一樣是 dataflow 跟 behavior，我想是因為算式只有一條，且不須透過 Full adder 就能計算，省了很多資源。

3. 最小 Delay 也沒意外是 dataflow 跟 behavior，我的想法跟上面一樣，因為算式只有一條，且不須透過 Full adder 就能計算，省了很

多資源。而 dataflow_reg 與 behavior_reg 因為多了一個 DFF，因此會慢一點。

4. 如果只看 fully combinational logic designs，因為 structure 是透過 full adder 相加起來，所以通過很多 Gate，因此數據是最差的。

5. 而加上 DFF 後，才會有 sequential area，我想是因為多了儲存的 register。

6. 波行的部分，在 RTL 與 Gate-level 長的一樣，我想是因為輸出結果想同，這也代表著驗證都正確。

六、心得

這次的作業感覺是為了使我們對所有工具熟悉一下，其實 RTL code 部分不會到很難，testbench 部分就比較需要思考，而如何去執行 logic synthesis 以及 pre/post simulation，都是我們比較不熟悉的，透過這次的作業，我認為已經對如何操作這些工具有一些心得了。

在 Testbench 的部分我遇到最多問題，一開始是完全不知道如何下手，甚至宣告 output 都不太會，不知道需不需要宣告 signed 等等，而我是透過不斷的 trial and error 得出最終的解答。