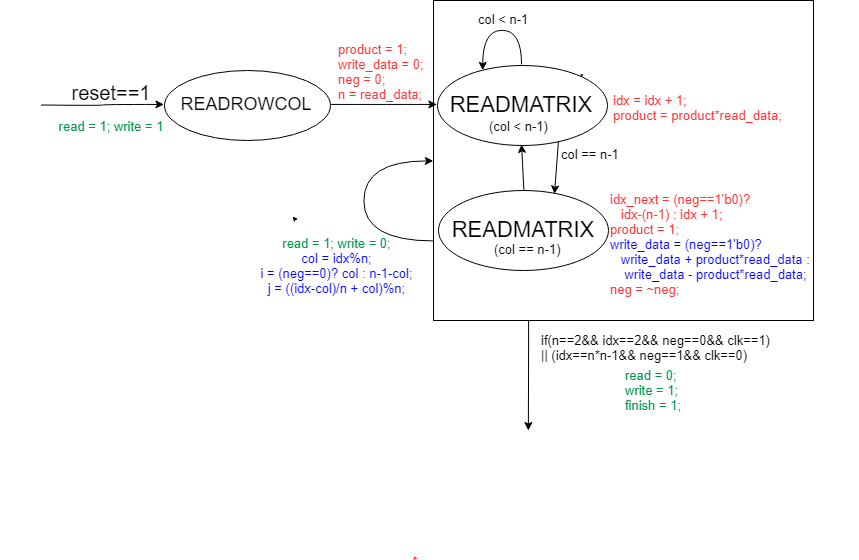
**Lab4 Bonus: Determinant**

# State Transition Graph



## Reason

有兩個state：READROWCOL和READMATRIX，但是READMATRIX中分為col < n-1和col == n-1的情況。

**READROWCOL**：將read、write都設為1，得到矩陣的行列數，記錄在n。到下一個state（READMATRIX），初始化product = 1， write\_data = 0，neg = 0。

**READMATIRX**：read為1，write為0，讀取矩陣。

* idx為目前累計讀取了幾次，通過idx可推算i與j（下面有說明）。
* product是目前這個cycle中得到的總積數，read\_data後與product相乘，更新product\_next。當col == n-1即完成1個cycle，write\_data就根據neg選擇加上或減掉這個cycle的積數。然後product歸零
* neg = ~neg：如果是加，拿下一個cycle就是減，反之亦然。

當idx==n\*n-1，neg = 1的時候輸出答案，立起finish。考慮到2×2矩陣的特殊情況：

(n==2'd2 && idx==2 && neg==1'b0 && clk==1'b1)

**col、idx、i、j、neg之間的關係**

例如在4×4矩陣，計算過程如下（矩陣Mij，括號內為idx）：

|  |  |  |  |
| --- | --- | --- | --- |
| M00 | M01 | M02 | M03 |
| M10 | M11 | M12 | M13 |
| M20 | M21 | M22 | M23 |
| M30 | M31 | M32 | M33 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| + | M00 (0) | \* | M11 (1) | \* | M22 (2) | \* | M33 (3) | Cycle 0 |
| - | M30 (0) | \* | M21 (1) | \* | M12 (2) | \* | M03 (3) | Cycle 1 |
| + | M01 (4) | \* | M12 (5) | \* | M23 (6) | \* | M30 (7) | Cycle 2 |
| - | M31 (4) | \* | M22 (5) | \* | M13 (6) | \* | M00 (7) | Cycle 3 |
| + | M02 (8) | \* | M13 (9) | \* | M20 (10) | \* | M31 (11) | Cycle 4 |
| - | M32 (8) | \* | M23 (9) | \* | M10 (10) | \* | M01 (11) | Cycle 5 |
| + | M03 (12) | \* | M10 (13) | \* | M21 (14) | \* | M32 (15) | Cycle 6 |
| - | M33 (12) | \* | M20 (13) | \* | M11 (14) | \* | M02 (15) | Cycle 7 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| col | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 |
| idx | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| i(neg==0) | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 |
| i(neg==1) | 3 | 2 | 1 | 0 | 3 | 2 | 1 | 0 | 3 | 2 | 1 | 0 | 3 | 2 | 1 | 0 |
| j | 0 | 1 | 2 | 3 | 1 | 2 | 3 | 0 | 2 | 3 | 0 | 1 | 3 | 0 | 1 | 2 |

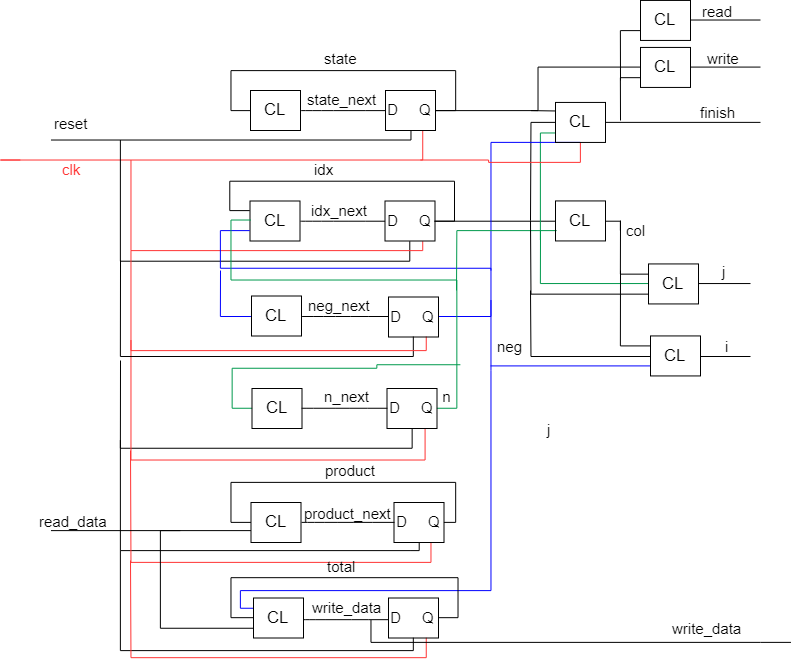
col、idx、i、j、neg關係如下表：

推算得知：

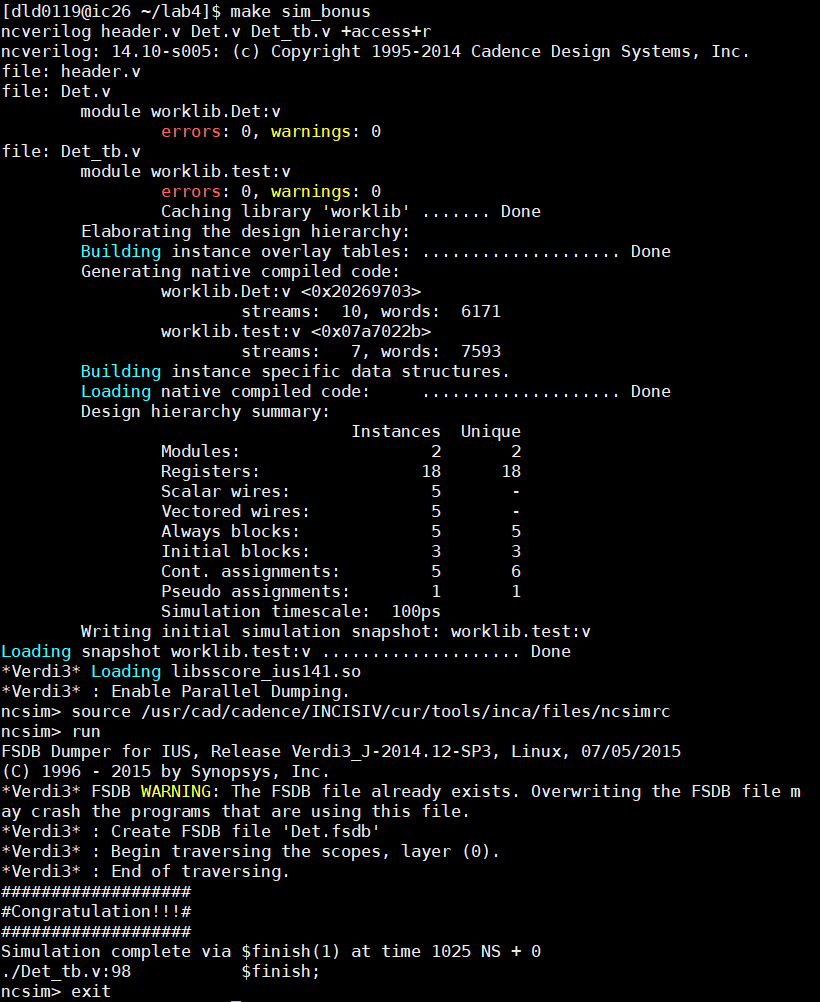
if(col<n-1) idx\_next = idx + 1;  
 else idx\_next = (neg==1'b0)? idx-(n-1) : idx + 1; *//如果剛才是做加法idx就回到該行的開始，做減法*  
 col = idx%n;  
    i = (neg==1'b0)? col : n-1-col;  
 j = ((idx-col)/n + col)%n;

\*這樣register只要有neg和idx就可以了，其他都是wire。

# Block Diagram



# ncverilog simulation (sim)



# ncverilog simulation (syn)

