2024 Digital IC Design

Homework 3: matrix multiplier

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| NAME | | 陳育政 | | | | | | |
| Student ID | | E24094198 | | | | | | |
| **Simulation Result** | | | | | | | | |
| Functional simulation | 100 | | Gate-level simulation | 100 | Clock  width | 17 ns | Gate-level simulation time | 98634 ns |
|  | | | | |  | | | |
| **Synthesis Result** | | | | | | | | |
| Total logic elements | | | | | 448 | | | |
| Total memory bit | | | | | 0 | | | |
| Embedded multiplier 9-bit element | | | | | 1 | | | |
|  | | | | | | | | |
| **Description of your design** | | | | | | | | |
| 我的設計是用分成四個states的FSM完成，state分別為MAT1\_READ、MAT2\_READ、MULTIPLY及OUTPUT。並且針對matrix1和matrix2，我是直接預設2個8-bit，固定大小為4\*4的2D array(我認為如果用變動大小的matrix size可能會導致在synthesis時有問題，因為就合成角度而言，要去合成一個會變動大小的硬體感覺並不合理)，並用row\_1、row\_2、col\_1及col­\_2控制in\_data要存到matrix的哪個位置，因此MAT1\_READ和MAT2\_READ的2個states寫法其實很相似。  至於invalid matrix會發生的原因，我有想到兩個可能性。第一個是matrix不同row輸入的column數不一，例如: matrix1的第一個row有3個components，但是第二個row卻變成有2個或4個components，則代表不是一個正常的matrix，至於第二個原因則是matrix1的column數和matrix2的row數不一致，導致兩個matrix無法相乘。關於第一個invalid matrix的偵測，我是多用一個MAT\_COL\_SIZE紀錄matrix的第一個row的column size，每當遇到col\_end signal被觸發時，就根據對應matrix檢查col\_1或col\_2和MAT\_COL\_SIZE的大小是否相等，若不相等就代表兩個row的column size並不一致，則把VALID\_MATRIX signal設為0。而第二個invalid matrix的偵測，則是直接在MULTIPLY state檢查col\_1是否等於row\_2即可。  再者，我發現助教提供的Scoring formula計算方式，multiplier的加權比重有達到9，因此為了降低這部分的消耗，我特別設計只用1個乘法器的module。假設matrix1\*matrix2=RESULT，在MULTIPLY state時，我會每次計算matrix1的一個cell和matrix2的一個cell相乘的結果值，並加到mat\_result這個register內，當我們乘完後，跳到OUTPUT state，把mat\_result的  值輸出到out\_data(mat\_result就是對應RESULT的其中一個cell的值)。  最後，在最初的模擬時，我發現in\_data這個訊號，會有延遲一個clock才有正確新的資料輸入，舉例來說: 假設上一輪最後in\_data的值是17，則當我要進到下一輪時，會把busy設回0，但是接下來的第一個in\_data仍會是17。 因此為了避免這樣的誤讀錯誤資料，我多設一個wait\_one\_cycle訊號，讓module在讀下一輪輸入前，多等一個cycle。 | | | | | | | | |

*Scoring = (Total logic elements + total memory bit + 9\*embedded multiplier 9-bit element) (Total cycle used\*clock width)*