2024 Digital IC Design

Homework 4: Max-Priority Queue

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| NAME | | 陳育政 | | | | | | |
| Student ID | | E24094198 | | | | | | |
| **Simulation Result** | | | | | | | | |
| Functional simulation | 100 | | Gate-level simulation | 100 | Clock  width | 30 ns | Gate-level simulation time | 3543 ns |
|  | | | | | A computer screen shot of a computer code  Description automatically generated | | | |
| **Synthesis Result** | | | | | | | | |
| Total logic elements | | | | | 15848 | | | |
| Total memory bit | | | | | 0 | | | |
| Embedded multiplier 9-bit element | | | | | 0 | | | |
| A screenshot of a computer  Description automatically generated | | | | | | | | |
| **Description of your design** | | | | | | | | |
| 這次的作業一開始我是直接把BUILD\_QUEUE寫在同一個state，但是在合成的時候發現似乎是BUILD\_QUEUE的邏輯太複雜，會讓合成的時間拉長很久，而且出來的critical path也會變長，因此後來我把BUILD\_QUEUE分成BUILD\_QUEUE\_1和BUILD\_QUEUE\_2兩部分實作。BUILD\_QUEUE\_1是決定largest的index，而BUILD\_QUEUE\_2則更新QUEUE的排序，這樣也的確大幅降低合成所需的時間。  至於最初register的宣告，我是在各個state各自宣告register，但後來我發現整個module中的單一個state最多只需要3個，因此便使用教授上課教過的resource sharing技巧把register共用，而我需要做的就只是在進入state之前，根據state所需先更新register的值。  在Post-Simulation的時候，我最初也沒有通過，而RAM\_D的輸出結果都是0，但是在Pre-Simulation又可以正常通過。後來我才發現不知為何我沒在reset階段初始化RAM\_A和RAM\_D會致使Quartus在合成的時候有一個警告，就是output stuck at VCC and GND，後果是合成的電路會讓RAM\_A和RAM\_D永遠卡在0或1，於是我把RAM\_A和RAM\_D初始化後，Post-Simulation的結果便成功通過了。  在實作Max-Priority Queue的演算法，我都是以助教提供的pseudo-code為主，而EXTRACT\_MAX和INSERT\_DATA因為在pseudo-code分別是直接呼叫BUILD\_QUEUE和INCREASE\_VALUE，於是在Verilog實作中，我是直接讓currState跳轉到對應的state，而非重複寫一次同樣的code。 | | | | | | | | |

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