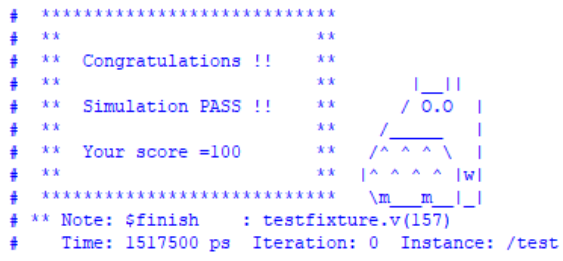
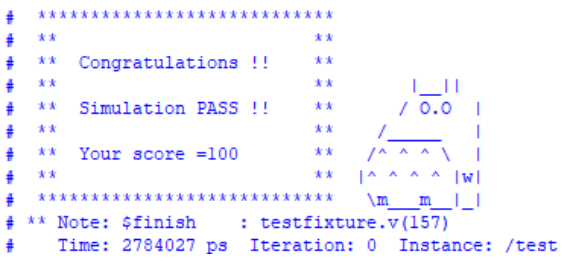
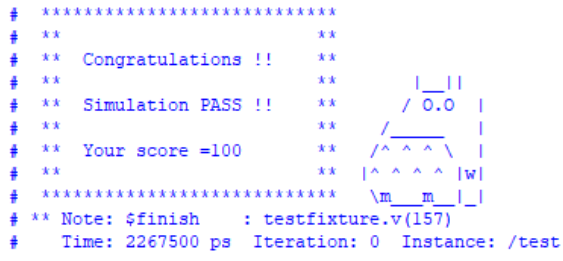
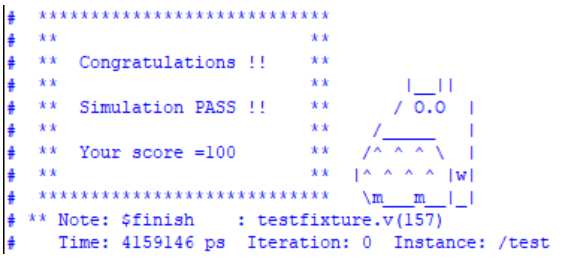
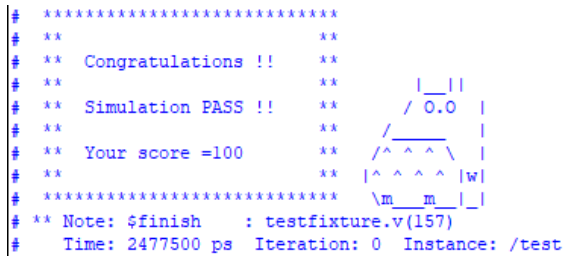
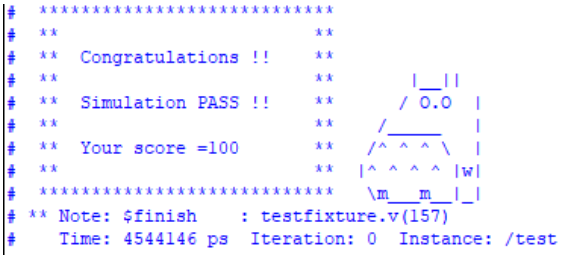
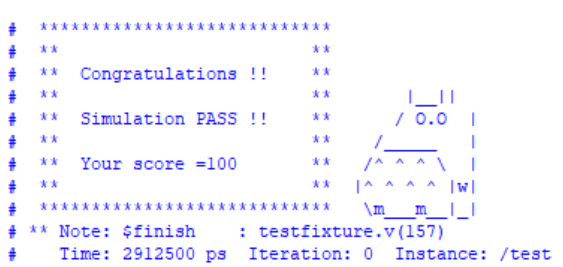
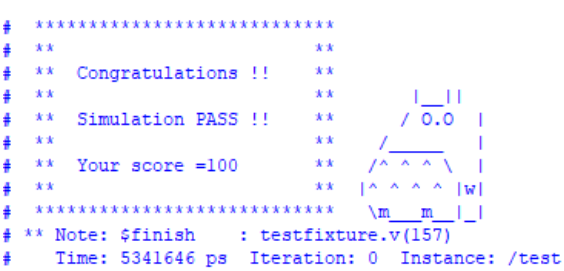
2024 Digital IC Design
Homework 4: Max-Priority Queue

NAME	陳俐蓉
Student ID	N26120113

Simulation Result

Functional simulation	P0: 100 P1: 100 P2: 100 P3: 100	Gate-level simulation	P0: 100 P1: 100 P2: 100 P3: 100	Clock width	27.5 ns	Gate-level simulation time	P0: 2784 ns P1: 4159 ns P2: 4544 ns P3: 5341 ns

your pre-sim result of test patterns	your post-sim result of test patterns
P0	P0
	
P1	P1
	
P2	P2
	
P3	P3
	

Synthesis Result	
Total logic elements	1203
Total memory bit	0
Embedded multiplier 9-bit element	0

your flow summary

Flow Status	Successful - Wed May 22 16:36:46 2024
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	MPQ
Top-level Entity Name	MPQ
Family	Cyclone IV E
Device	EP4CE55F23A7
Timing Models	Final
Total logic elements	1,203 / 55,856 (2 %)
Total registers	206
Total pins	50 / 325 (15 %)
Total virtual pins	0
Total memory bits	0 / 2,396,160 (0 %)
Embedded Multiplier 9-bit elements	0 / 308 (0 %)
Total PLLs	0 / 4 (0 %)

Description of your design

本設計目的為實現 Max-Priority Queue，根據輸入指令執行 Build_Queue、Extract_Max、Increase_Value、Insert_Data、Write_RAM 四種功能，過程中需要維持 max heap tree 的資料結構，也就是 parent node 必須大於 child node 的 complete tree。

依功能分成 9 個模組，5~9 的模組會根據不同的指令被啟動功能，因此都是使用狀態機控制，主要分成三個狀態: **IDLE**、**EXCTE**、**FINISH**，**IDLE** 時等待被啟動，啟動後進入 **EXCTE** 開始執行功能，執行完畢後進入 **FINISH**，下個 cycle 再回到 **IDLE**。

1. MPQ
   Top module，負責子模組接線，包含其他 8 個模組。

2. Controller
   負責控制整個 MPQ 的狀態機，分為四個狀態: **IDLE**、**INPUT_DATA**、**INPUT_CMD**、**EXCUTE**，初始狀態為 **IDLE**，輸入資料的 valid 訊號拉起後，進入 **INPUT_DATA** 開始將 tree 的資料存入暫存器，資料輸入完畢後進入 **INPUT_CMD**，表示可以開始接收指令，接收一個指令後進入 **EXCUTE**，開始執行指令的功能，執行完畢後，如果此指令為 Write_RAM，則回到 **IDLE** 等

待下筆 tree 的輸入，否則回到 **INPUT_CMD** 繼續接收下個指令。

3. Data_reg
   負責暫存 max heap tree 的資料，並根據不同指令的功能去調整資料的 index 的資料。

4. Max_Heapify
   負責使 tree 的 parent 大於 left child 和 right child，用交換資料的 while 迴圈來實現，狀態機分為 6 個狀態:
   - **IDLE:** 等待被啟動，啟動時 largest index 更新為 root index，並且下個狀態進入 **COMP_L**。
   - **UPDATE_RL:** 更新 left index (= largest index*2)和 right index (= largest index*2+1)的值，下個狀態進入 **COMP_L**。
   - **COMP_L:** 如果 left 的資料大於 largest 的資料，將 largest index 更新為 left index，下個狀態進入 **COMP_R**。
   - **COMP_R:** 如果 right 的資料大於 largest 的資料，將 largest index 更新為 right index，下個狀態進入 **SWAP**。
   - **SWAP:** 如果 largest index 有被更新過(largest index!= root index)，則交換 root 和 largest 的資料，使 root 的資料比 left 和 right 大，並將 root index 更新為 largest index，接著下個 cycle 回到 **UPDATE_RL**，否則進入 **FINISH**。
   - **FINISH:** 整個 tree 的資料交換完畢，下個 cycle 回到 **IDLE**。

5. Build_Queue
   輸入指令為 000 時會啟動此模組，負責維持 tree 為 max heap tree 的資料結構，讓每個 parent node 都大於 child node，當接著此模組會去啟動 Max_Heapify，直到每個 parent node 都完成檢查，接著狀態會進入 **FINISH**，通知 Controller 可以繼續接收下個指令。

6. Extract_Max
   當輸入指令為 001 時會啟動此模組，負責將 tree 中最大的值(tree 目前的 root)移除，過程中需要保持 max heap tree 的資料結構，所以會去啟動一次 Max_Heapify。

7. Increase_value
   當輸入指令為 010 時會啟動此模組，會將輸入 index 的資料更新為輸入的 value，如果更新後的值大於 parent，就會交換 parent 和此 index 的資料，直到將 tree 重新調整為 max heap tree 的資料結構。

8. Insert_Data
   當輸入指令為 011 時會啟動此模組，負責加入新的資料，會先將 tree 的 size 加 1，並且將最大 index 的資料設為 0，接著啟動 Increase_value，將最大 index 的資料更新為輸入的 value，並在過程中保持 max heap tree 的資料結構。

9. Write_RAM
   當輸入指令為 100 時會啟動此模組，負責將執行完前面指令的 tree 資料寫入 RAM 中，讓 testbench 比對是否正確。

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