2025 Digital IC Design Homework 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NAME | 黃啟桓 | | | | |
| Student ID | P76134927 | | | | |
| **ATCONV Simulation Result** | | | | | |
| Functional simulation | | Pass | | Pre-Layout simulation | Pass |
| https://cdn.discordapp.com/attachments/778196329250816000/1374646267409596526/image.png?ex=68380919&is=6836b799&hm=371002a22b67dfb75803cf15f8f7408d83849a1838664ac3cd2495847bf27cfd&= | | | | https://cdn.discordapp.com/attachments/778196329250816000/1374992345916899369/image.png?ex=6838a2a9&is=68375129&hm=16260f8af1346ac11bd8912f4e9bc91326c3ae8d63571ff7944bf1614a5d3bdb&= | |
| **System Simulation Result** | | | | | |
| Functional simulation | | Pass | | Pre-Layout simulation | Fail |
| https://cdn.discordapp.com/attachments/778196329250816000/1377363805998284820/image-52.png?ex=6838b181&is=68376001&hm=0caeb28e96c077e3726e527519b83cc5689cd1cf5a50fb4f7c522280a5799427&= | | | | 沒結果 | |
| **ATCONV Synthesis Result** | | | | | |
| Total logic elements | | | 28500 | | |
| Total memory bits | | | 0 | | |
| Total registers | | | 10216 | | |
| Embedded multiplier 9-bit elements | | | 0 | | |
| Total Cycle used | | | 4233 //合成需要合半小時，執行需要半小時 | | |
| https://cdn.discordapp.com/attachments/778196329250816000/1374693510409879612/image.png?ex=68383519&is=6836e399&hm=5206cf327f8edde0a0159dc7f68eee1ff8306c275a08eba8f7c4f600b15fadb4&= | | | | | |
| **System Synthesis Result** | | | | | |
| Total logic elements | | | 無 | | |
| Total memory bits | | | 無 | | |
| Total registers | | | 無 | | |
| Embedded multiplier 9-bit elements | | | 無 | | |
| Total Cycle used | | | 無 | | |
| 無 | | | | | |
| **Description of your design** | | | | | |
| 當初就是朝著"如何做得更快"的想法去實現，所以特別壓縮計算Layer0、Layer1的時。  輸入的頻率最快就是1個cycle，所以output也不用去增加pipeline去多組同步運算，反而是當layer0 1個cycle計算1個值比1個cycle計算多個值更優，因為Layer 0 可以更快開始算，更快的輸出。Layer 1可以更快的開始算更快的輸出。  時間上: read\_raw = 4096 cycle, calc\_L0 = 4096 cycle, calc\_L1 = 1024 cycle, start\_calc\_L0 = 130 cycle(in read\_raw)  已知read\_raw, calc\_L0, calc\_L1可以同時進行  所以時間上最佳結果就是 4096 + 130 cycle (+ rst甚麼時候開始) = 4226  ============================  儲存資料的結構，類似使用cycle queue的方式，原本raw array需要開68\*68，但實在不需要，所以就用64\*64去推論，接著每一次計算L0 最多只需要5行，所以有效raw\_array就保留5\*64，當然為了方便運算，採用8\*64，這樣就能直接用shift，時間上來說肯定更快  每次輸入會先覆蓋掉最先前的data因為他們確實已經沒用了。  不過等我意識到register array開的大小會影響到logic element時，我已經來不及了，不想改了，所以就勉強用用  ============================  System 的設計其實是延續原版，但不同的是bus最多一次一個資料傳輸，又考慮到其實計算L0 L1很快，所以不用在讀取raw array時，提前計算L0或L1。  然後步驟就很清晰了  1. 讀取raw 2行。(這時還不能開始算L0)  2. 讀取raw 2行。  3. 算L0 2行  4. 算L1 2行 (回到state 2，但如果raw以讀取完就回到state 3) | | | | | |