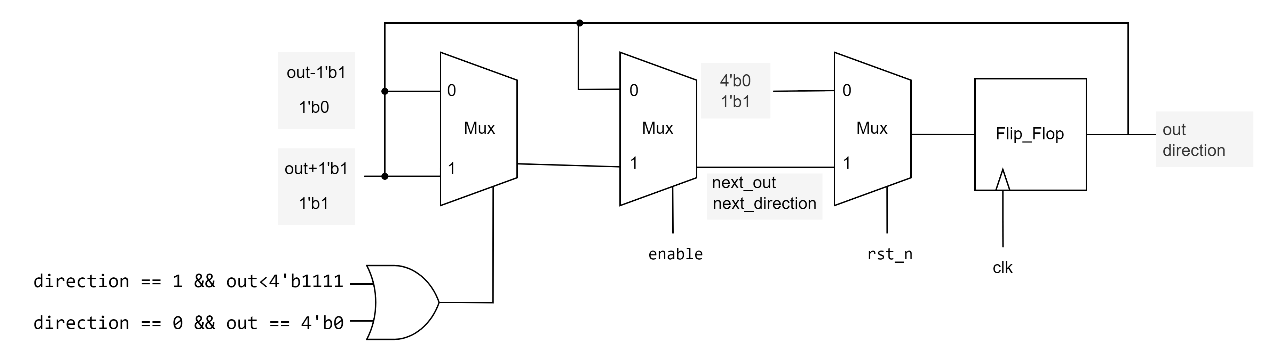
Lab 3 report

組員：110062221李品萱

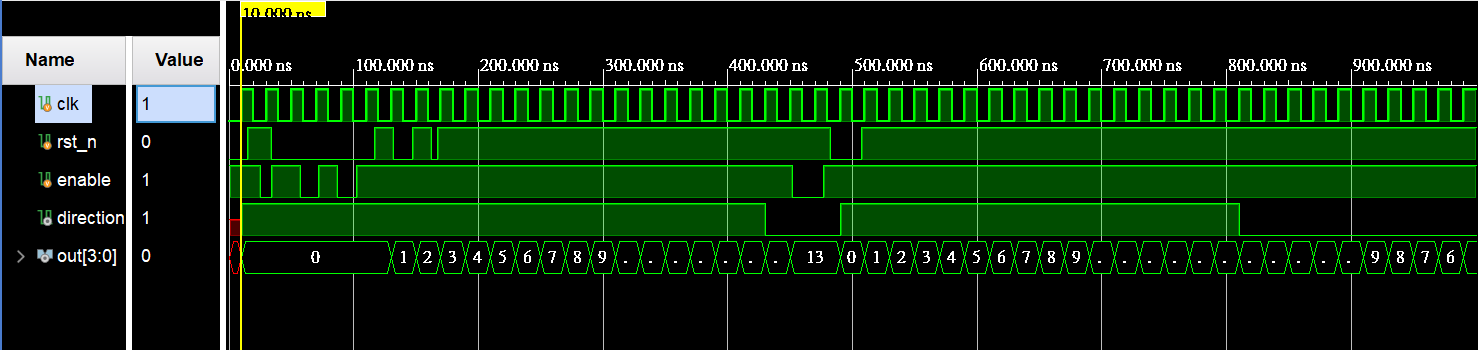
110062213唐翊雯

1. **4-bit Ping-Pong Counter**

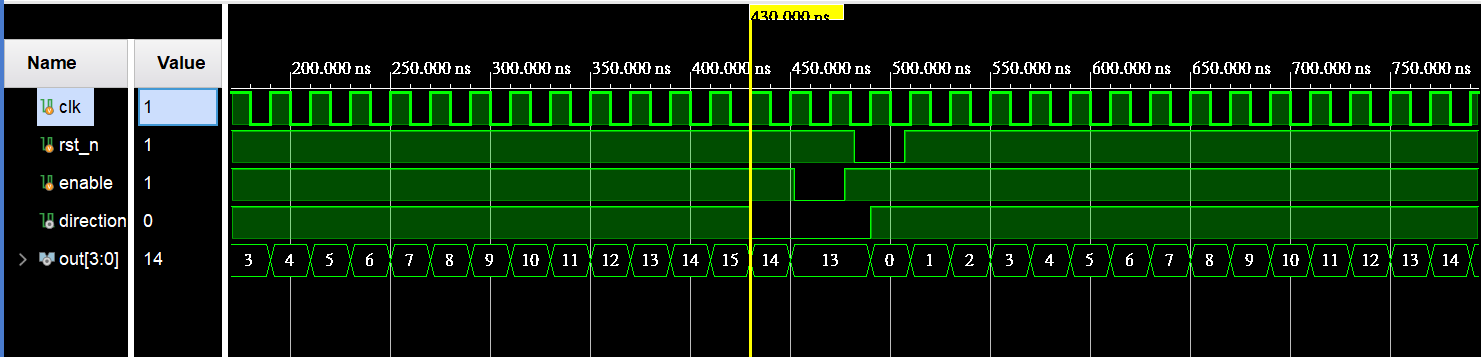
如下圖，這題我們需要實作一個Ping-Pong Counter，首先在combinational block的部份我們先用next\_out及next\_direction接下一次clock起來的時候要傳過Flip Flop的值。為了讓這個Counter counting up時能做到Spec中數到15後將direction從1變為0及counting down數到0後將direction從0變為1，我們觀察到只有Counting up的out在0到14的區間內或Counting down的out為0時，會使next\_direction = 1’b1，next\_out = out+1’b1，因此我們用if-else去做判斷，而它synthesize的結果即為Mux。接下來我們的next\_direction及next\_out會再經過用enable作為select訊號的Mux判斷是否要維持當前的out及direction，若enable為1，我們再進到下一個Mux用rst\_n判斷是否將out設為0及direction設為1，而enable及rst\_n我們都是在sequential block內去做設計，因為它們都是在posedge clk時才去判斷這次的out 及direction分別是什麼，簡單來說，combinational block會負責處理next\_direction及next\_out的值，而sequential block則負責配合clock處理最後的out及direction。



在下面的波型圖中我們可以看到，在clk起來時才會trigger到rst\_n為0，並將我們的counter設定初始值，在一開始會造成一段時間out為0及direction為1的情形是因為我們對(rst\_n, enable) = (0, 1), (1,0), (0, 0)做測試，且以上情況都會造成上述的結果。



接下來我們在Counter運算中進行測試，如下圖，首先在out為15時，此時direction為1，而下一個clock時direction即為0並開始往下數。在out為13後經過一個clock我們讓enable為0，此時如預期的會使counter hold value，而後我們讓rst\_n為0，Counter也確實使out回到0，direction回到1。



1. **First-In First Out (FIFO) Queue**
2. **Multi-Bank Memory**
3. **Round-Robin FIFO Arbiter**
4. **4-bit Parameterized Ping-Pong Counter**
5. **4-bit Parameterized Ping-Pong Counter on FPGA**