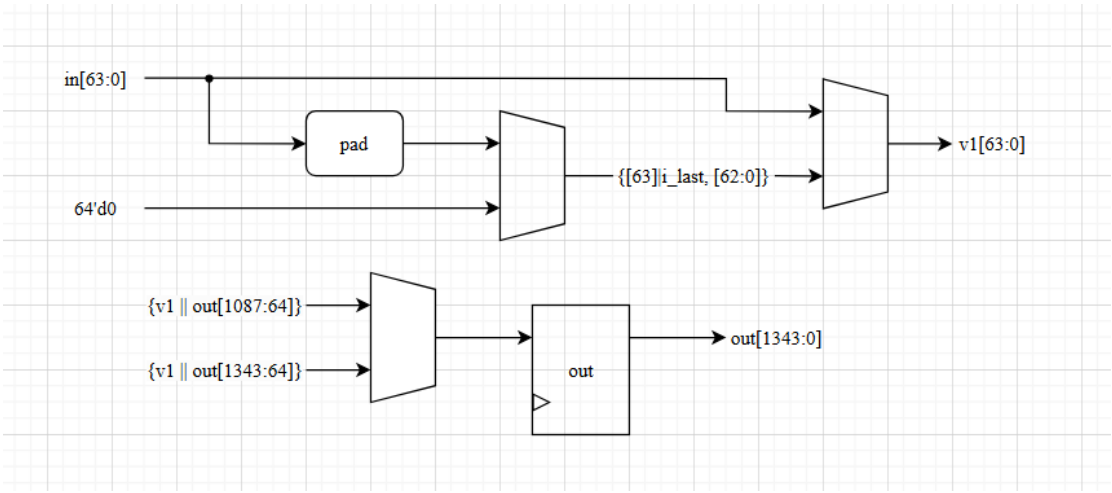
Keccak

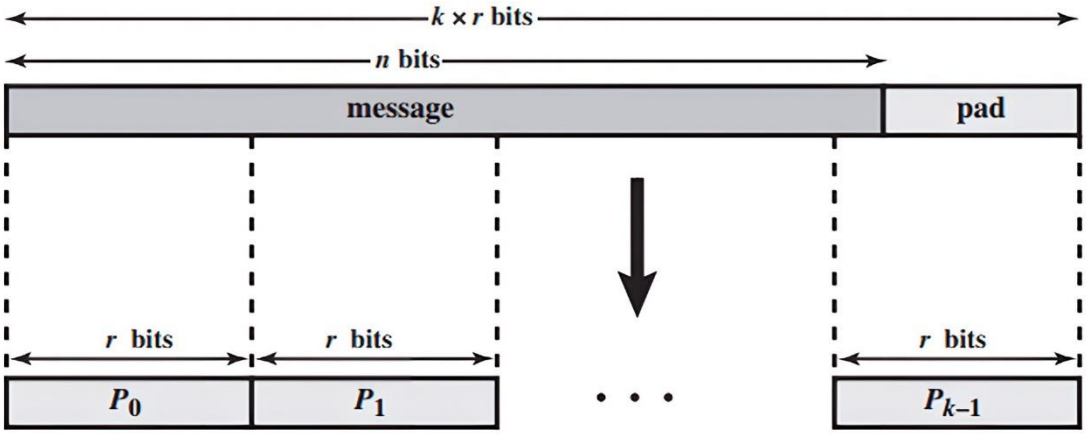
輸入為 64bit，模式有 G 跟 H 兩種，分別為 shake128 與 shake256，在 sha3 電路設計上我分為兩個主要的 module，一個是 Padder，專門處理資料輸入進來、shake 需額外 padding 的 4'd1111 以及進行 Multi-rate padding 的部分，另外一個是 F_Permutation，專門處理 Sponge Construction 的部分。

SHA3 primitives	MLDSA primitives	Padding	Size in bits		
			r	c	output length
SHAKE128	G	M || 0x1f || 0x00 … || 0x80	1344	256	unlimited
SHAKE256	H	M || 0x1f || 0x00 … || 0x80	1088	512	unlimited

1. Padder

每次輸入近來的資料長度為 64bit，要 padding 一筆資料會分為 3 種情況，第一種就是一般的資料輸入，直接將輸入的 64bit 送進輸出的位移暫存器，而第二種是該筆 64bit 資料為剛好為 1088 或者 1344 的最後一個 64bit 輸入，則會將輸入送入 pad 當中並且將他的最後一位拉至 1，最後一個情況是輸入的有效資料已經送完，但是還須補 64'd0 至 1088 或者 1344，如果補的 64'd0 是最後一筆則將他的最後一位拉至 1。當組合好 1088 或 1344 時，則會送出 out_ready 信號告訴 F_Permutation 可以動作。如果 F_Permutation 正在運算且 1344bit 的暫存器被存滿則會透過 buffer_full 告訴外界無法接收資料。





2. F_Permutation

   當 Padder 完成後，會透過 in_ready 告訴 F_Permutation 可以開始執行 round function，最一開始進來會根據模式不同與 out 暫存器中的資料做 xor，接者開始進行 24round 的 round function，因為一次的 round function 是很長的組合電路，會成為整體電路的 critical path，因此我根據路經 delay 長度將每次 round functoin 拆為兩段，第一次做 theta、rho、pi，第二次做 chi、iota，完成後會向 Keccak 送出 out_ready 信號，如果輸出的有效資料不夠用，則可透過 squeeze 信號，讓 F_Permutation 再產生有效資料。

Function	Type
θ	Substitution
ρ	Permutation
π	Permutation
χ	Substitution
ι	Substitution