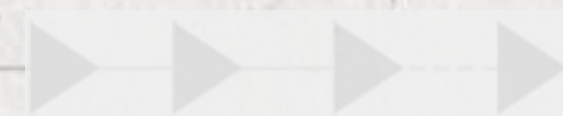
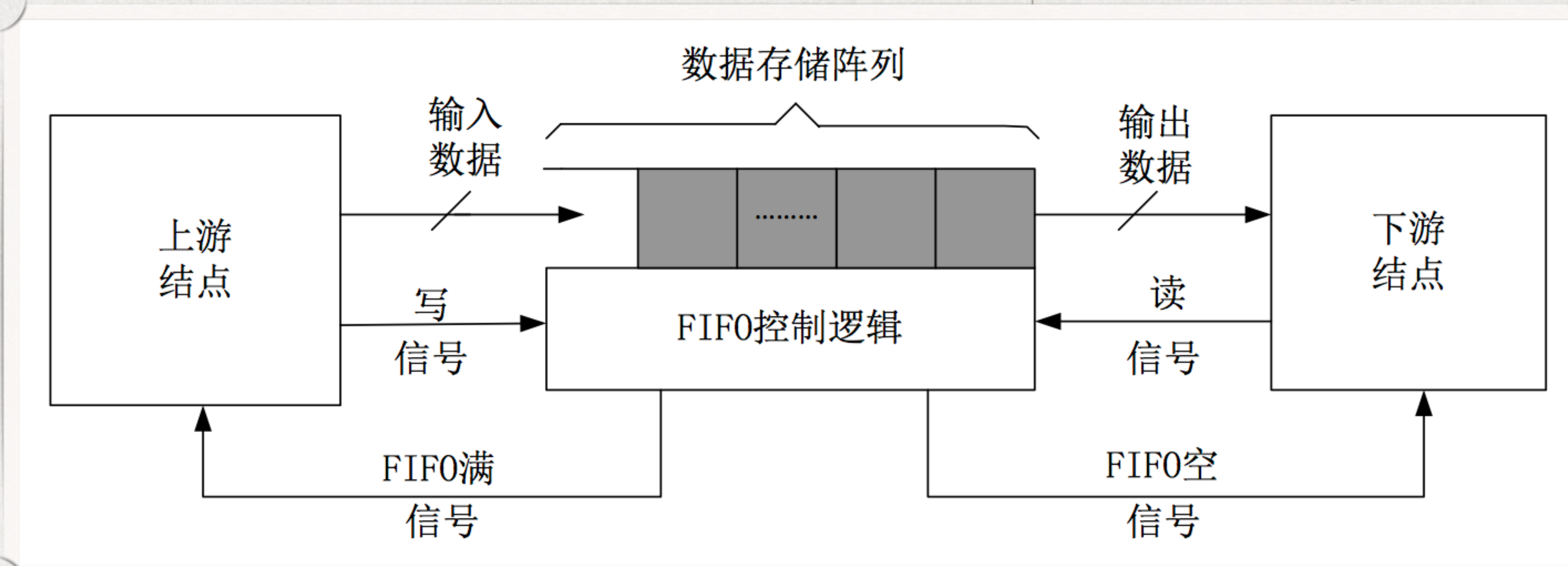


芯动力——硬件加速设计方法

第三章 同步电路与跨时钟域电路设计(3)

邸志雄@西南交通大学
zxdi@home.swjtu.edu.cn





- FIFO的上游结点是FIFO的数据输入端，在写信号有效时，数据将被写入FIFO的顶部（由FIFO内部的写指针控制），并且在FIFO内部，写指针后移一个单元，同时FIFO的满信号（FIFO full Signal）将控制上游结点是否发送数据；FIFO的下游节点是FIFO的数据输出端，当读信号有效时，FIFO中位于FIFO底部单元的数据将被读出（由FIFO内部的读指针控制），并且在FIFO内部读指针将后移一个单元，同时FIFO空信号（FIFO empty Signal）将控制下游节点是否读出数据。

常见参数

FIFO的宽度:

FIFO的深度:

满标志:

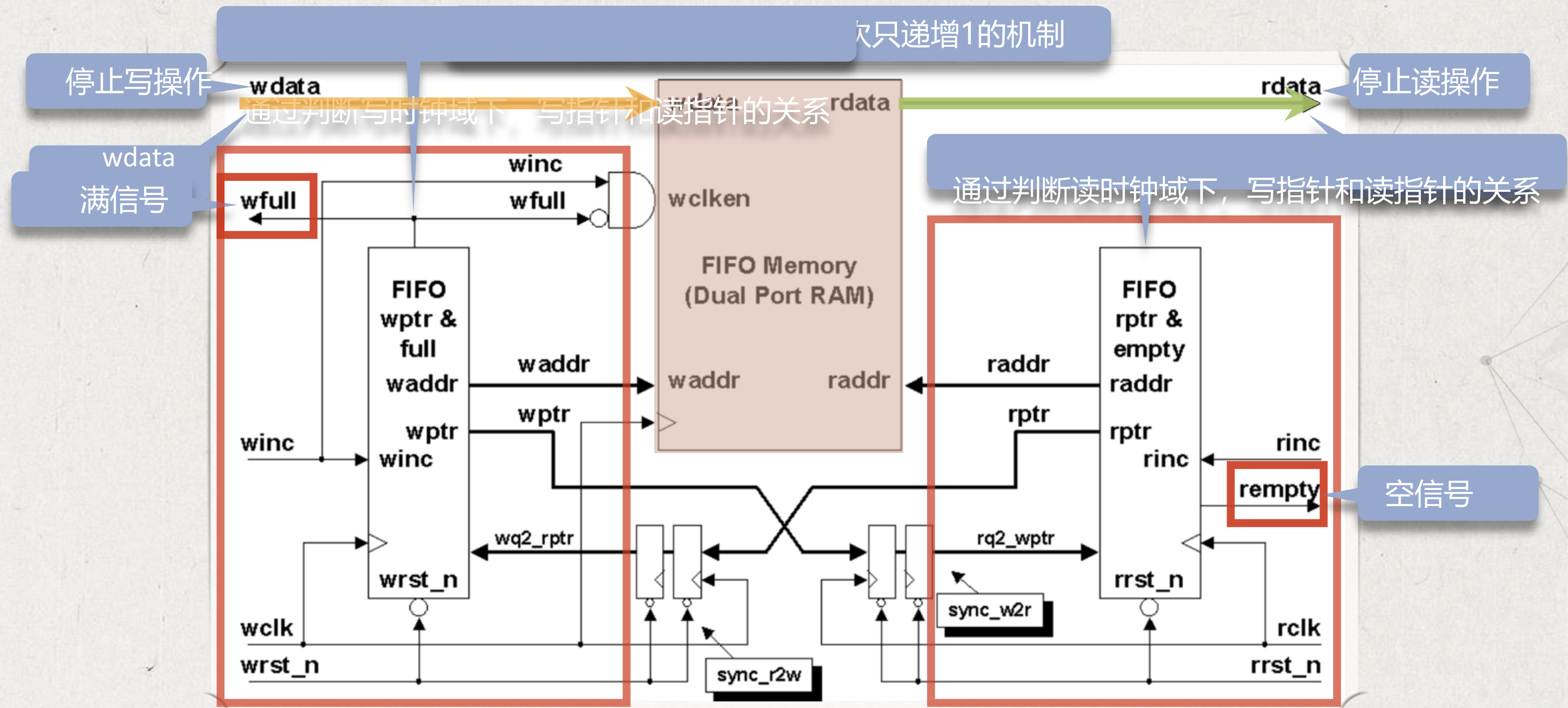
空标志:

读时钟:

写时钟:

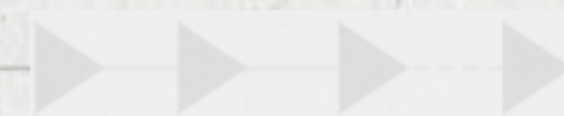
同步策略 — FIFO

- 即FIFO一次读写操作的数据位;
- 是FIFO可以存储多少个N位的数据 (如果宽度为N) 。
- FIFO已满或将要满时由FIFO的状态电路送出的一个信号, 以阻止FIFO的写操作继续向FIFO中写数据而造成溢出 (overflow) 。
- FIFO已空或将要空时由FIFO的状态电路送出的一个信号, 以阻止FIFO的读操作继续从FIFO中读出数据而造成无效数据的读出 (underflow) 。
- 读操作所遵循的时钟, 在每个时钟沿来临时读数据。
- 写操作所遵循的时钟, 在每个时钟沿来临时写数据。



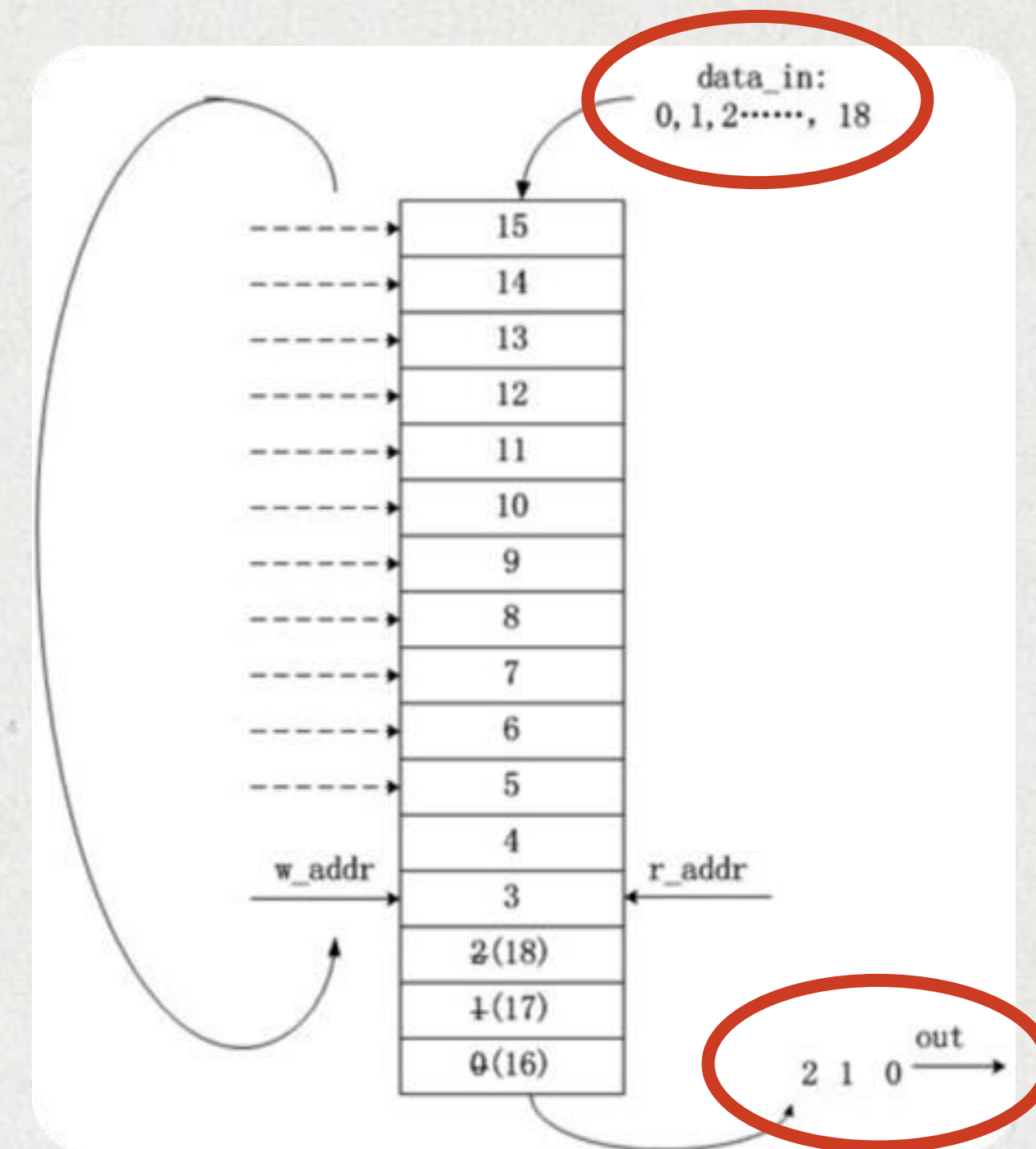
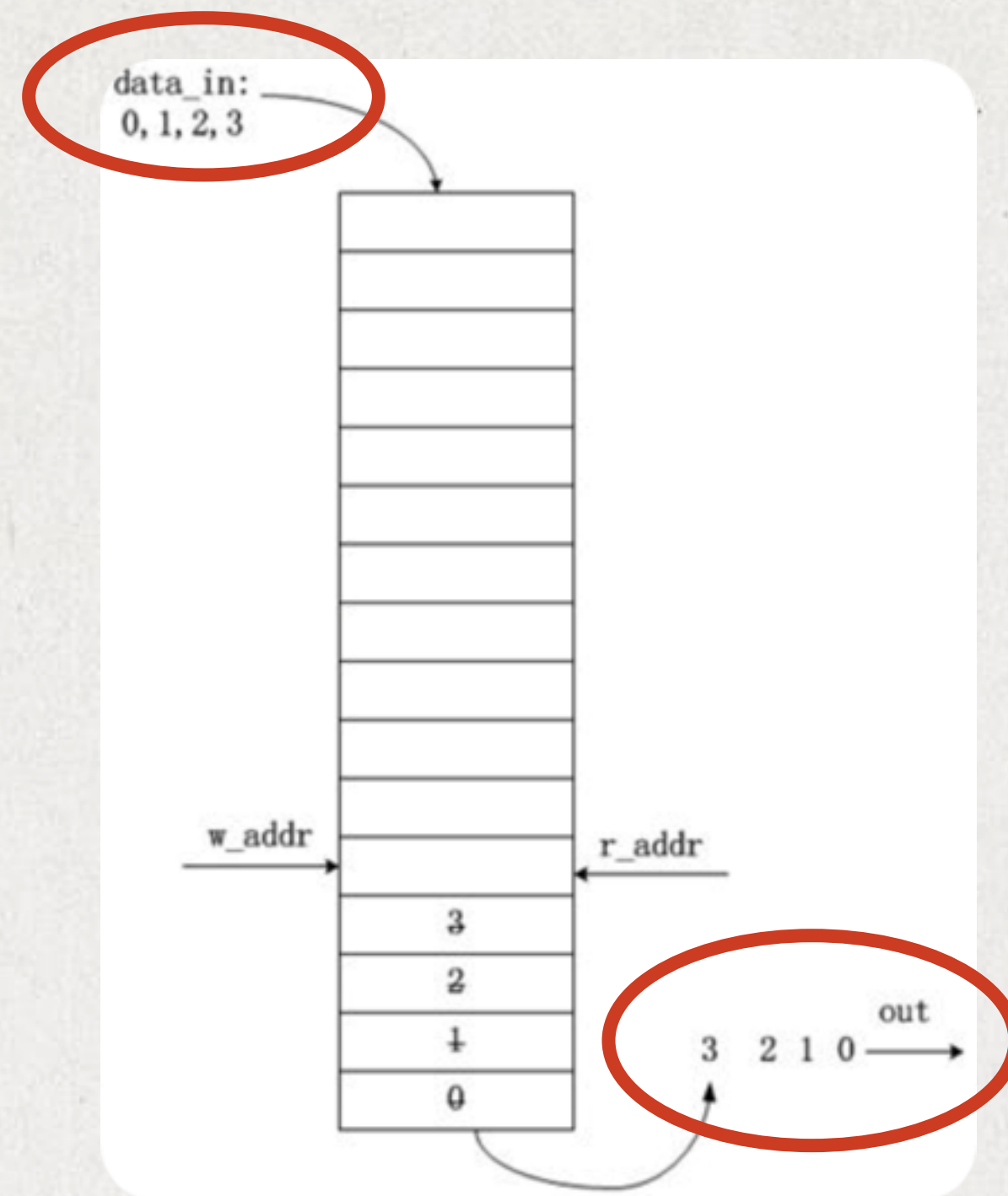
关键:

- 亚稳态的消除;
- 空满状态的判断;



空满状态的判断

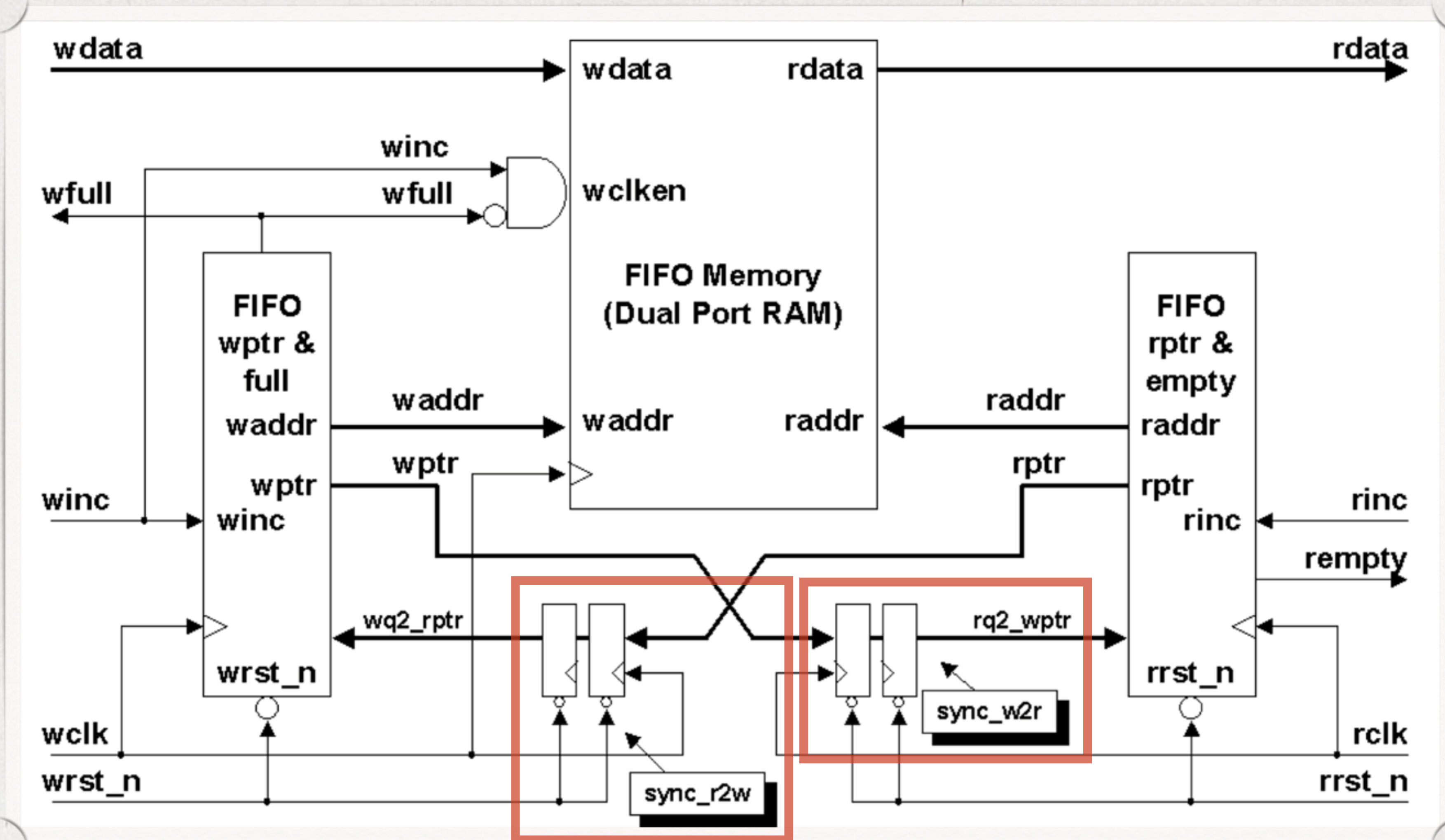
- 当读写指针相等时，表明FIFO为空，这种情况发生在复位操作时，或者当读指针读出FIFO中最后一个字后，追赶上了写指针；



- 当读写指针再次相等时，表明FIFO为满，这种情况发生在，当写指针转了一圈，折回来(wrapped around)又追上了读指针。

为了区分到底是满状态还是空状态，可以采用以下方法：

- 在地址中添加一个额外的位(extra bit)，当写指针增加并越过最后一个FIFO地址时，就将写指针这个未用的MSB加1，其它位回零。对读指针也进行同样的操作。此时，对于深度为 2^n 的FIFO，需要的读/写指针位宽为 $(n+1)$ 位，如对于深度为8的FIFO，需要采用4bit的计数器，0000 ~ 1000、1001 ~ 1111，MSB作为折回标志位，而低3位作为地址指针。
- 如果两个指针的MSB不同，说明写指针比读指针多折回了一次；如 $r_addr=0000$ ，而 $w_addr = 1000$ ，为满。
- 如果两个指针的MSB相同，则说明两个指针折回的次数相等。其余位相等，说明FIFO为空。



格雷码的特点:

- 格雷码相邻的2个数值之间只会有一位发生变化, 其余各位都相同。
- 格雷码是一种循环码, 0和最大数($2^n - 1$)之间也只会有一位不同。

数值	二进制码	格雷码
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

引用格雷码之后, 相邻数值只有1位发生翻转, 1位翻转所引起的亚稳态的概率远远要小于几位同时翻转所引起的概率; 因此, 格雷码能很好的亚稳态出现的概率。

转化方法：

- 二进制码转化为格雷码：从最右边第一位开始，依次将每一位与左邻一位异或(XOR)，作为对应格雷码该位的值，最左边一位不变；
- 格雷码转化为二进制码：从左边第二位起，将每位与左边一位解码后的值异或(XOR)，作为该位解码后的值（最左边一位依然不变）。

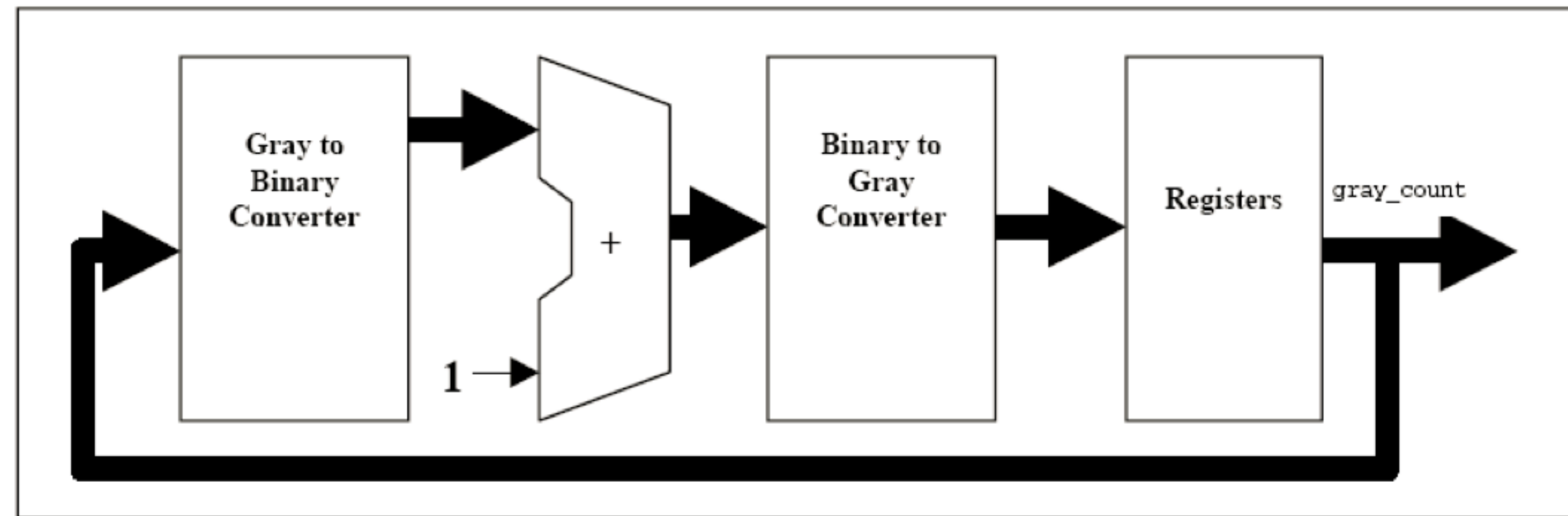
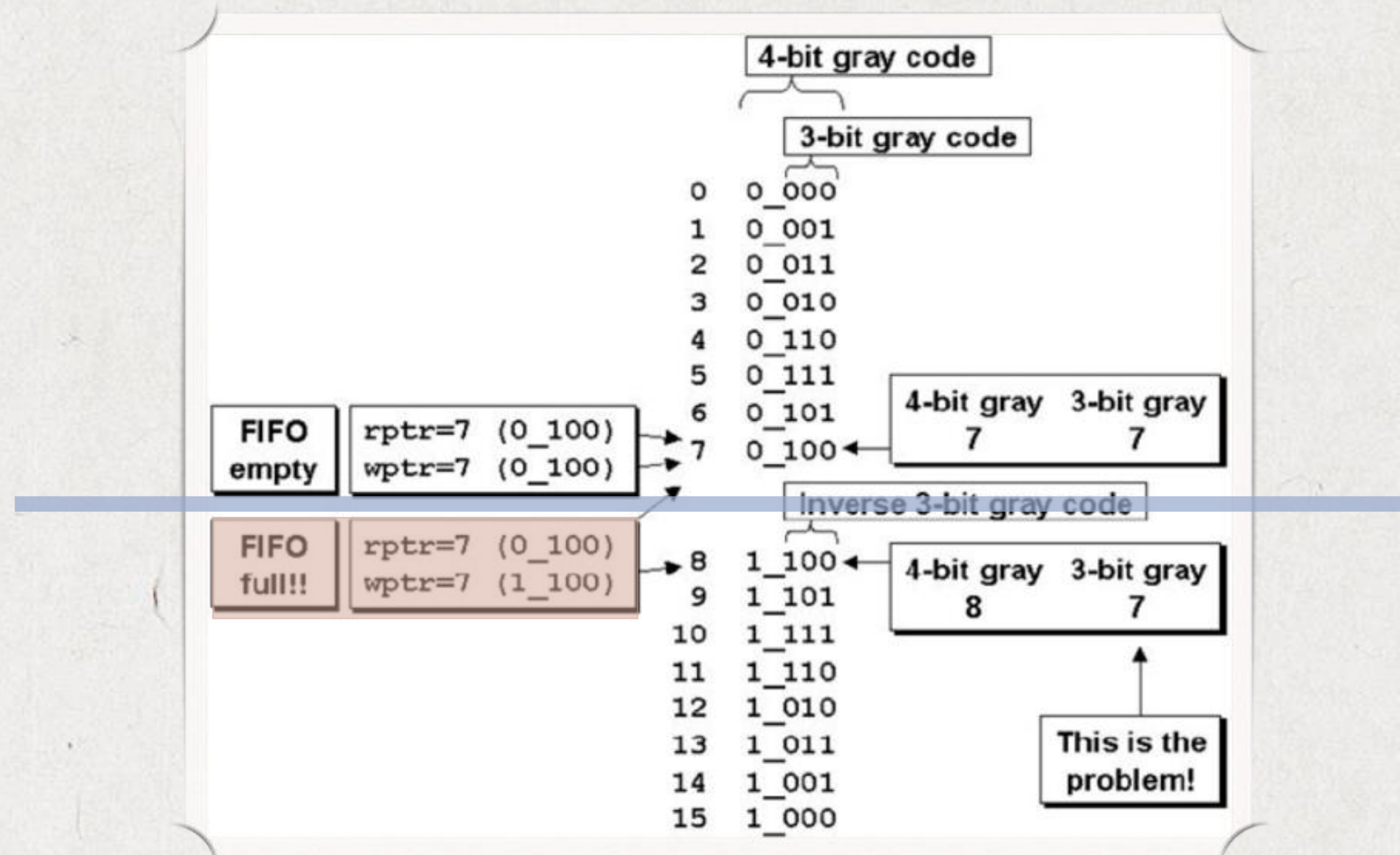


Figure 3 Generalized Gray counter architecture

使用gray码进行对比，如何判断“空”与“满”？



空

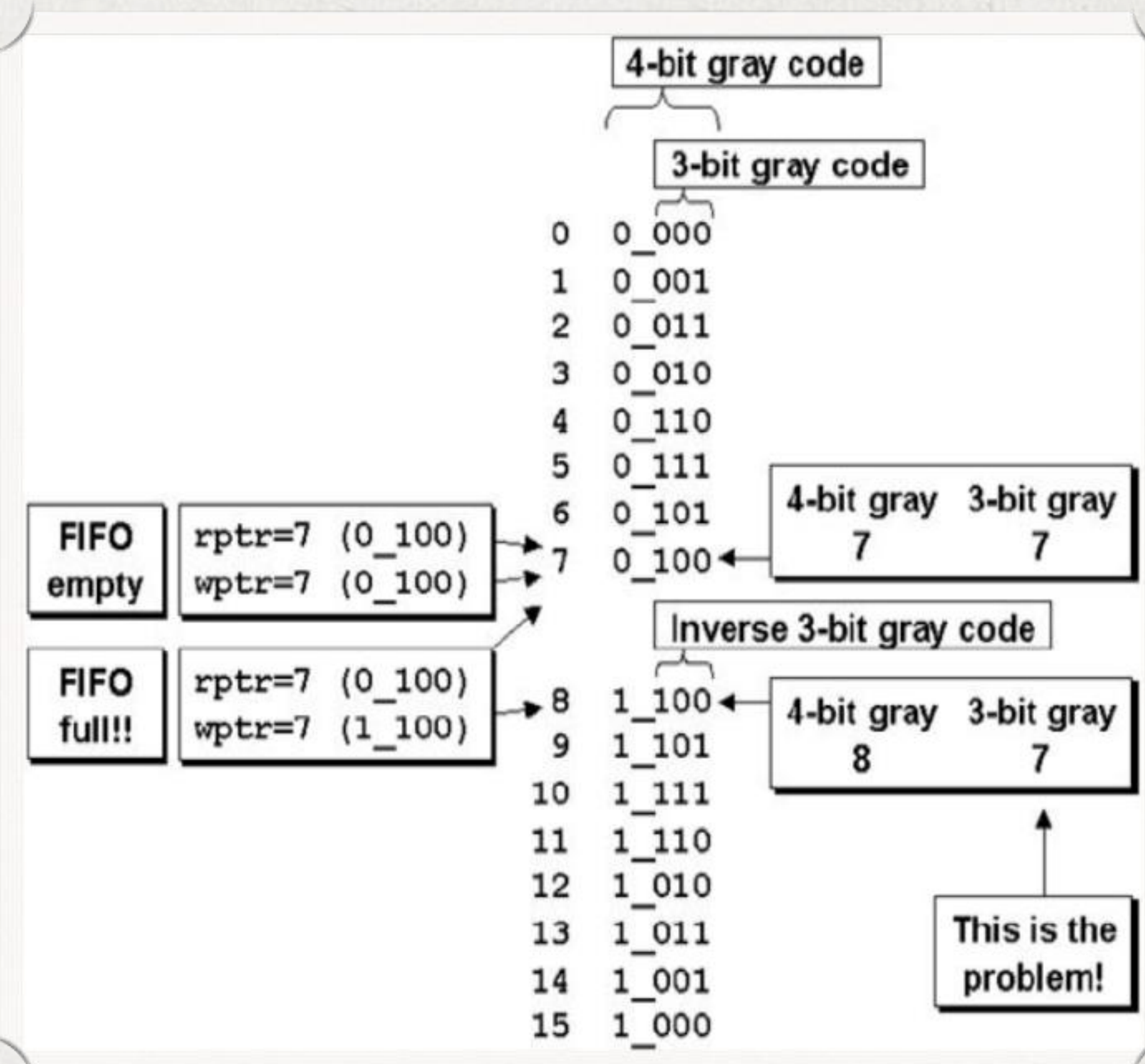
- 依据二者完全相等(包括MSB)

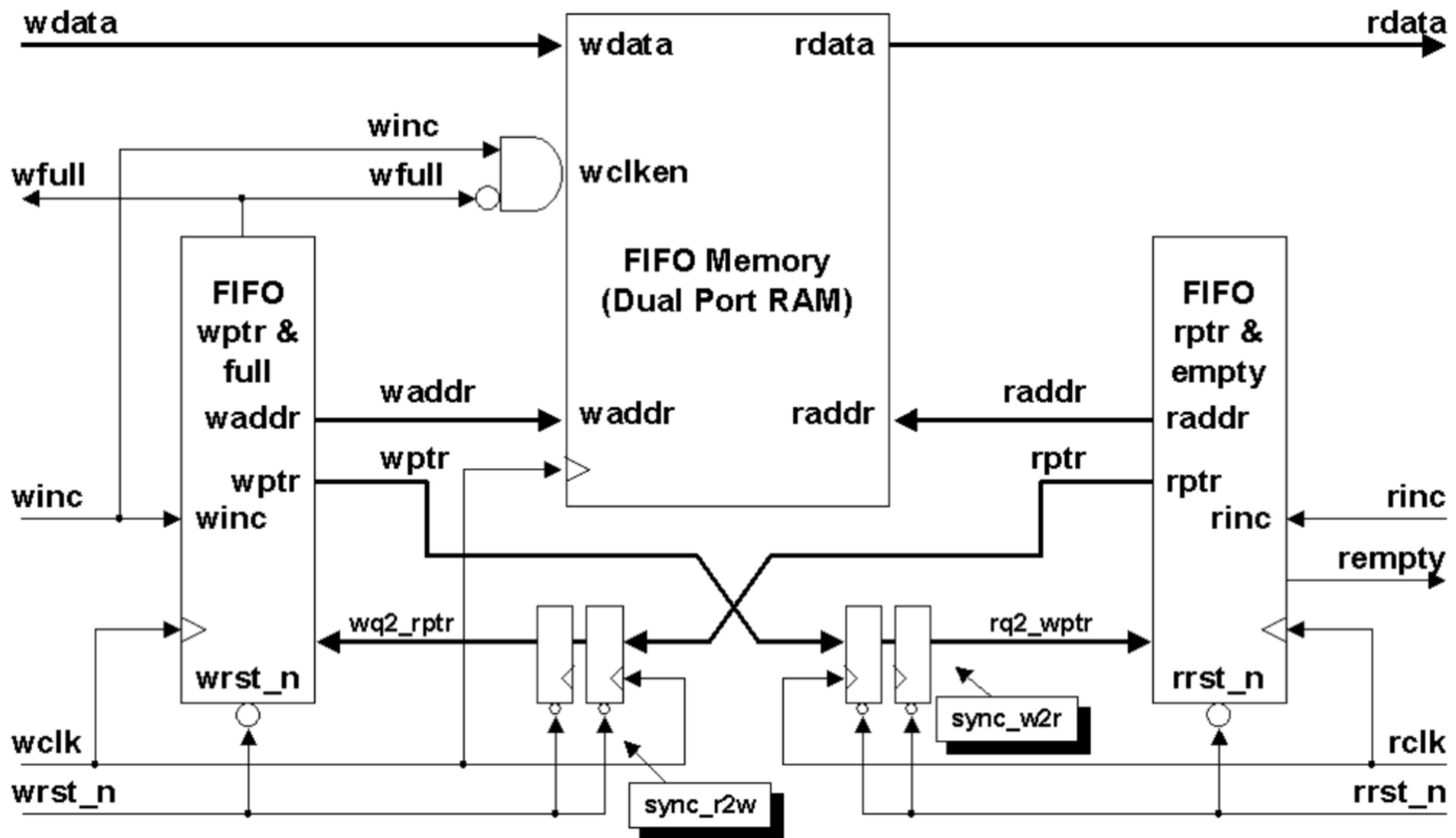
满

- 由于gray码除了MSB外，具有镜像对称的特点，当读指针指向7，写指针指向8时，除了MSB，其余位皆相同，不能说它为满。

在gray码上判断为满必须同时满足以下3条：

- wptr和同步过来的rptr的MSB不相等，因为wptr必须比rptr多折回一次。
- wptr与rptr的次高位不相等，如上图位置7和位置15，转化为二进制对应的是0111和1111，MSB不同说明多折回一次，111相同代表同一位置。
- 剩下的其余位完全相等。



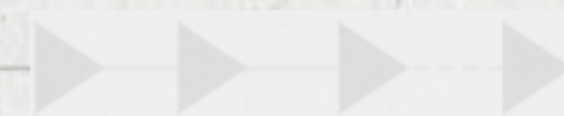


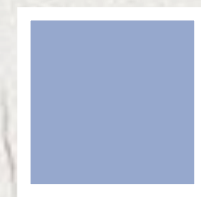
FIFO深度

例题：

- 假设FIFO的写时钟为100MHz，读时钟为80MHz。在FIFO输入侧，每100个写时钟，写入80个数据；读数据侧，假定每个时钟读走一个数据。

请问FIFO深度设置多少可以保证FIFO不会上溢出和下溢出？

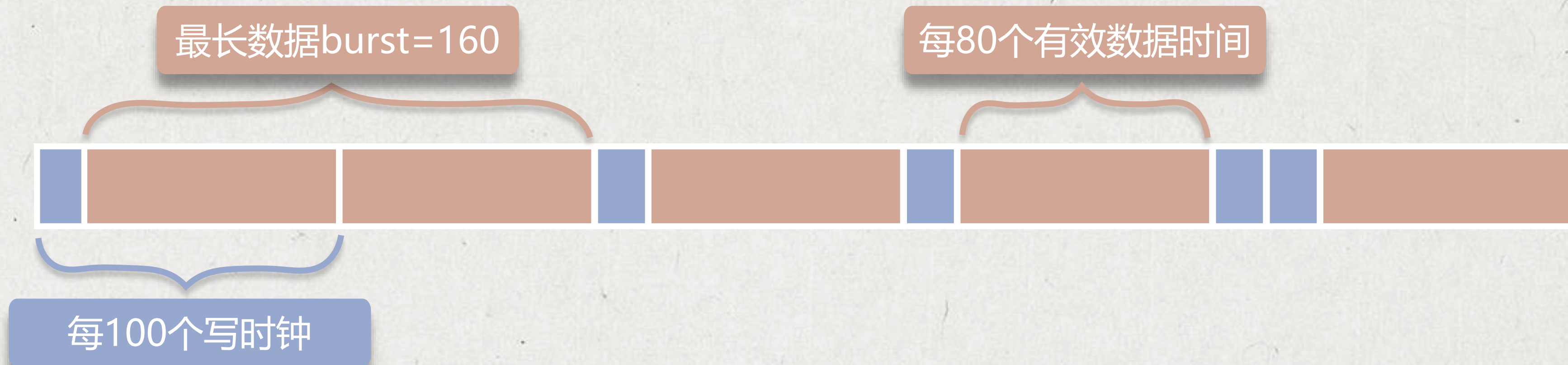




无数据写入

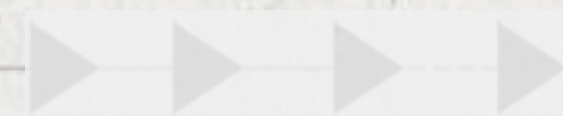


有数据写入



解: • 假设写入时为最坏情况（背靠背），即在 $160 \times (1/100)$ 微秒内写入160个数据。以下是写入160个burst数据的时间计算方法：

$$Time : burst_cycle \times t_{wa} = \frac{burst_cycle}{f_{clk}} = \frac{160}{100}$$

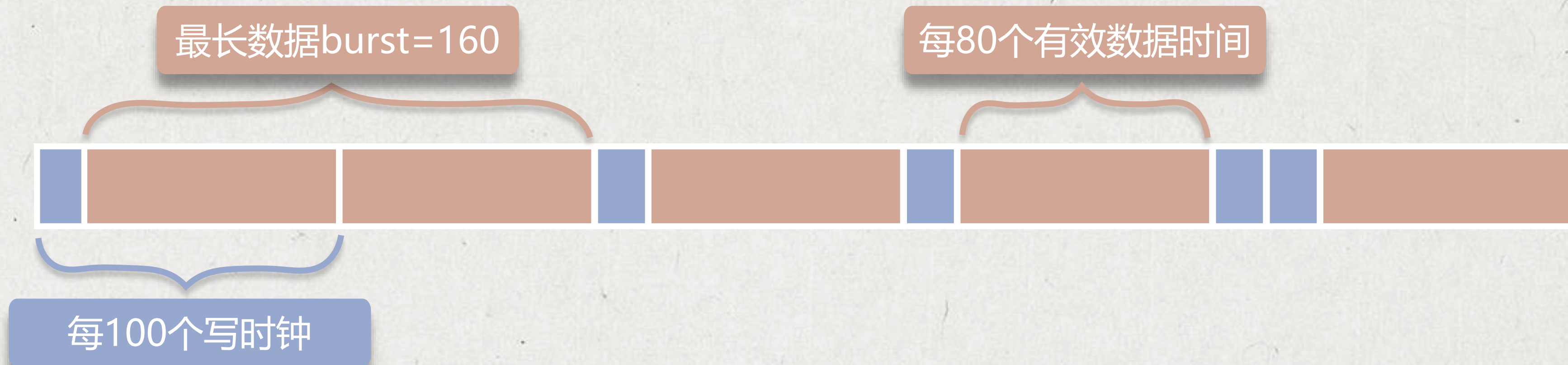




无数据写入

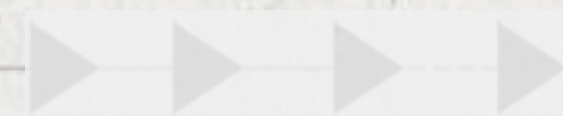


有数据写入



解: • 在这段时间内只能读出 $160 \times (1/100) \times 80$ 个数据

$$data_num_{read} = \frac{Time}{t_r} = \frac{Time}{\frac{1}{f_r}} = Time \times f_r = \frac{160}{100} \times 80$$

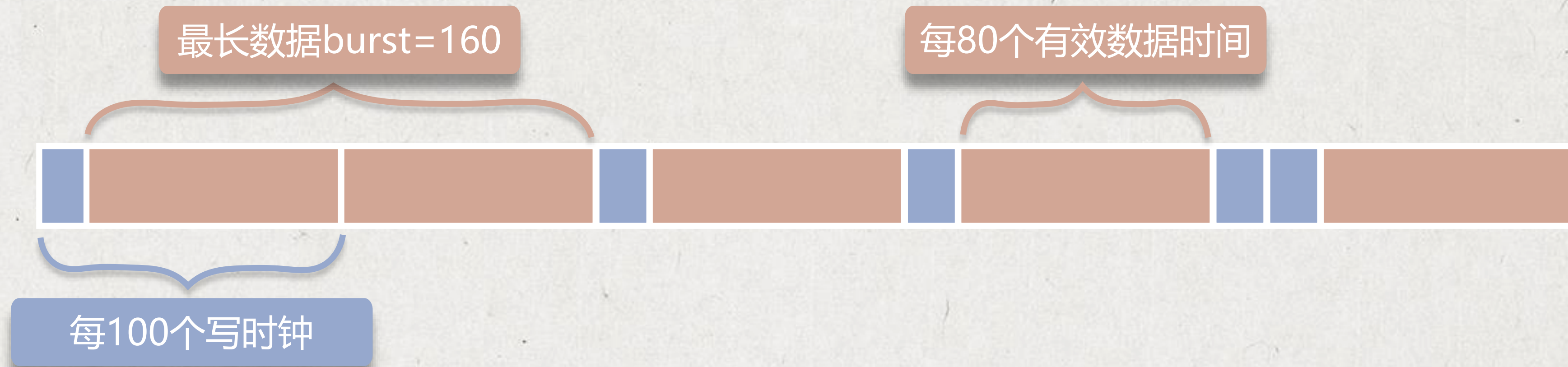




无数据写入



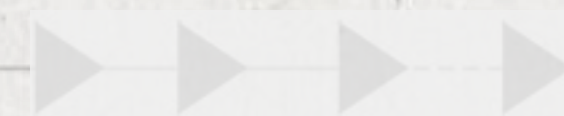
有数据写入



解:

$$depth = burst - data_numread = 160 - \frac{160}{100} \times 80 = 32$$


- 因此该FIFO深度为32



将问题一般化:

- 写时钟频率: $WCLK$;
- 读时钟频率: $RCLK$;
- 写时每 B 个时钟周期内会有 A 个数据写入FIFO;
- 读时每 Y 个时钟周期内会有 X 个数据读出FIFO;




$$depth = burst_length - \frac{burst_length}{wclk} \times \left(rclk \times \frac{X}{Y} \right)$$

- 其中(burst_length/WCLK)表示这个burst的持续时间，(RCLK*(X/Y))表示读的实际速度，两者的乘积自然就是这段时间读出的数据量。
- burst_length表示这段时间写入的数据量。
- 写入和读出两者之差为FIFO中残留的数据，这个也就是理论上的FIFO的最小深度。

