Asynchronous FIFOs

Pointer Exchange

- Each side needs to know what memory address the other is pointing to in order to determine Full (write domain) or Empty (read domain)
- These pointers could be exchanged using a handshake, but that would be required on every exchange and have a large performance impact
- Instead, Gray pointers are exchanged
- Parallel, 2 flip-flop synchronizers may then be used, since only one bit will change for any pointer update

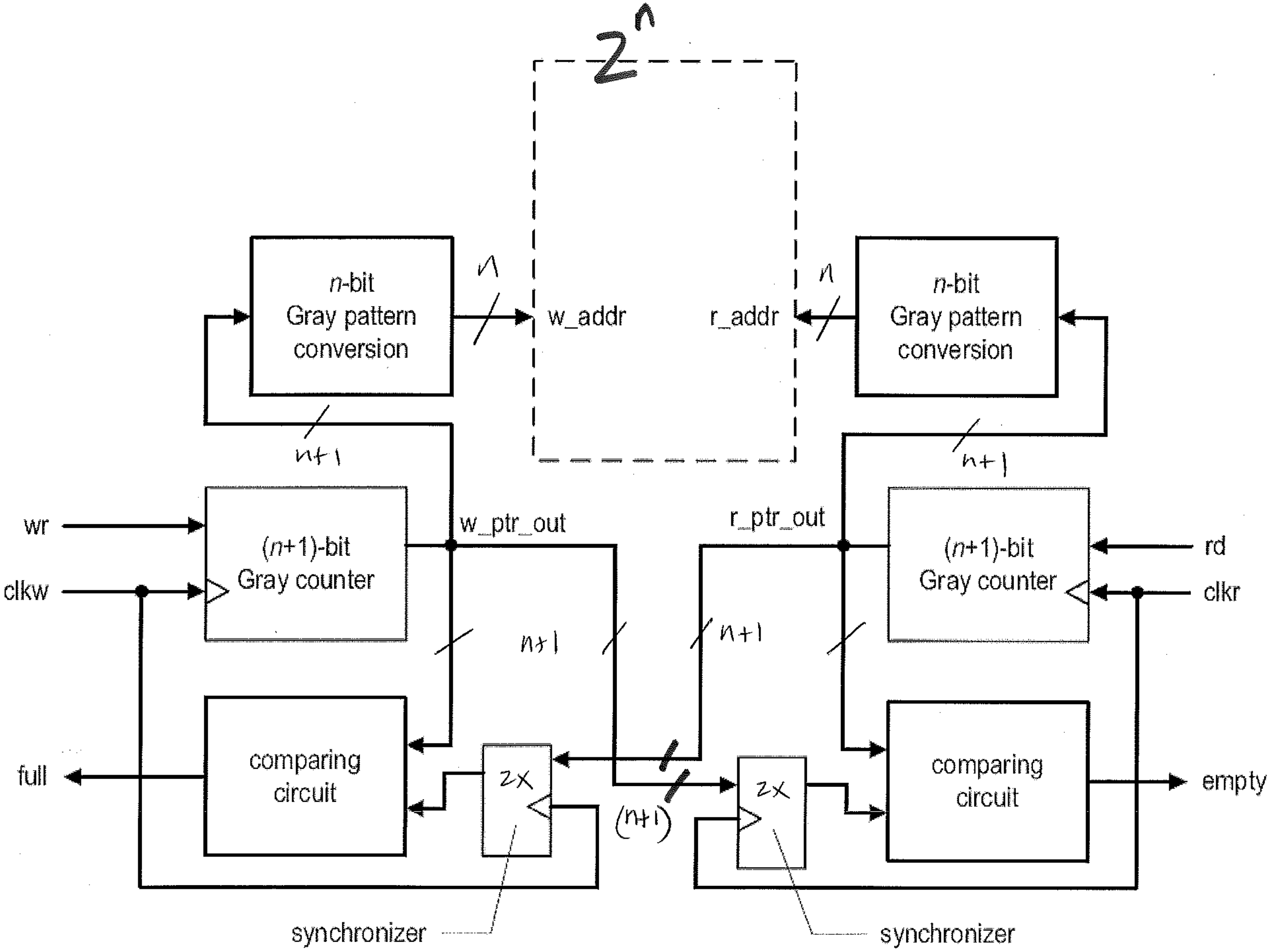
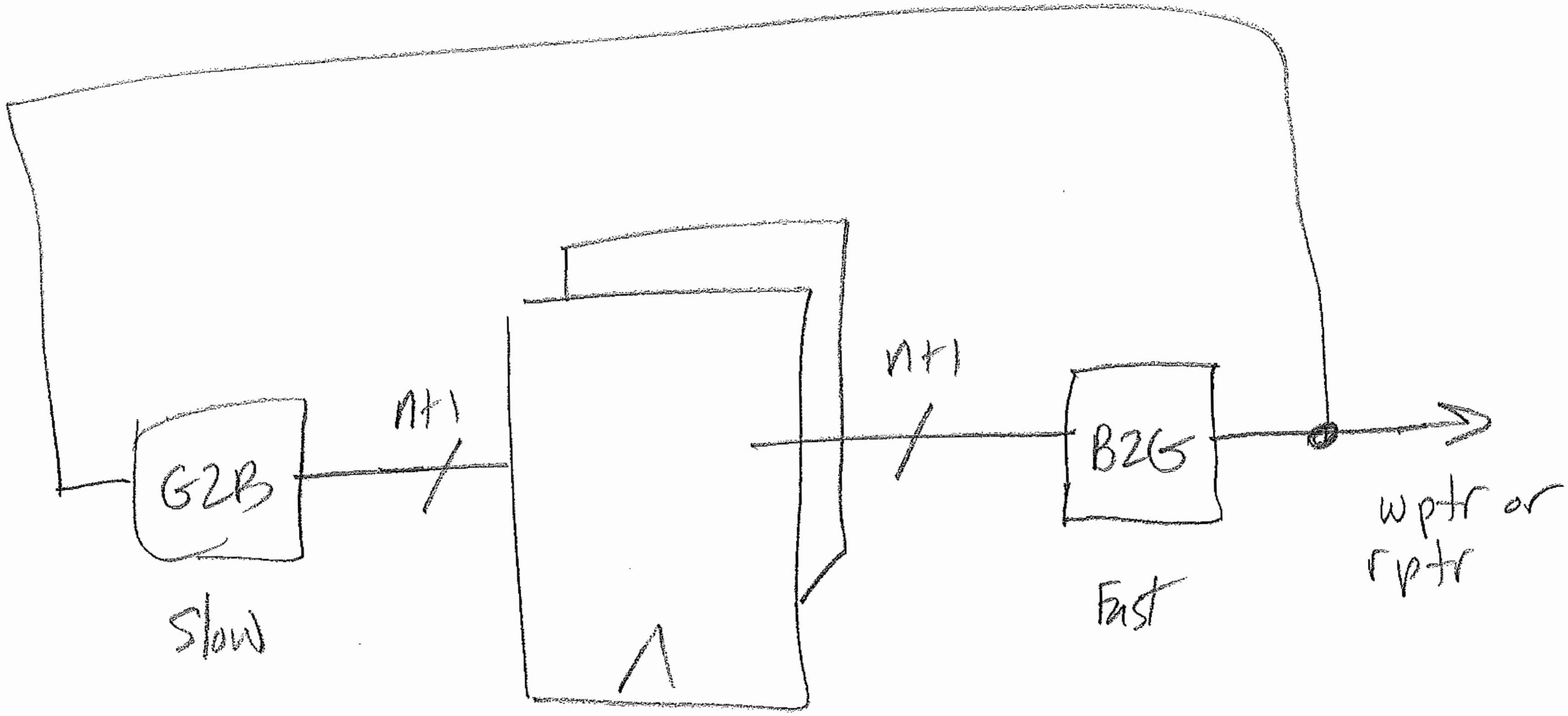

2^n
n-bit
Gray pattern
conversion
n
w_addr
r_addr
n
n-bit
Gray pattern
conversion
n+1
n+1
wr
clkw
(n+1)-bit
Gray counter
w_ptr_out
r_ptr_out
(n+1)-bit
Gray counter
rd
clkr
n+1
n+1
comparing
circuit
full
2x
(n+1)
2x
comparing
circuit
empty
synchronizer
synchronizer

Fig 16,26

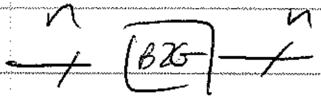
Gray Code Counters

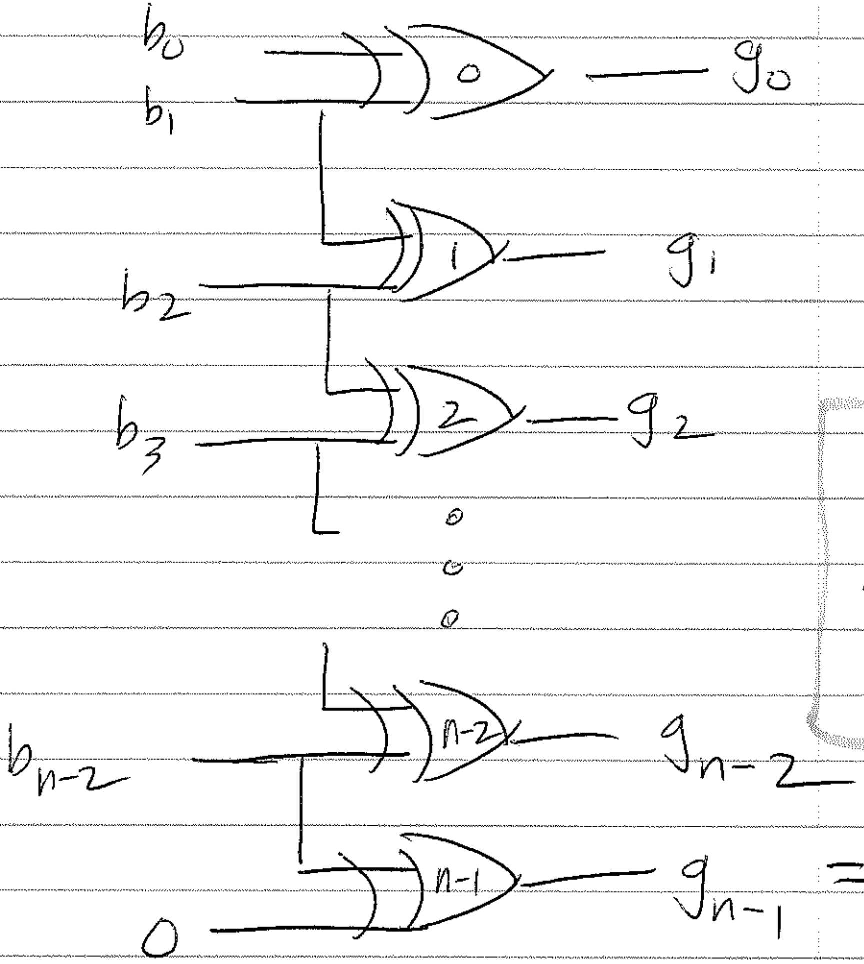
- There is no way to count in Gray code, so the method presented in class used a binary adder and conversion circuits to convert the binary code to Gray code and then back to binary for incrementing
- Both conversions require approximately n XOR2 gates for an n-bit pointer
- The conversion from binary to Gray can be done in parallel, but the other must be done serially, increasing the propagation delay

Gray Counter



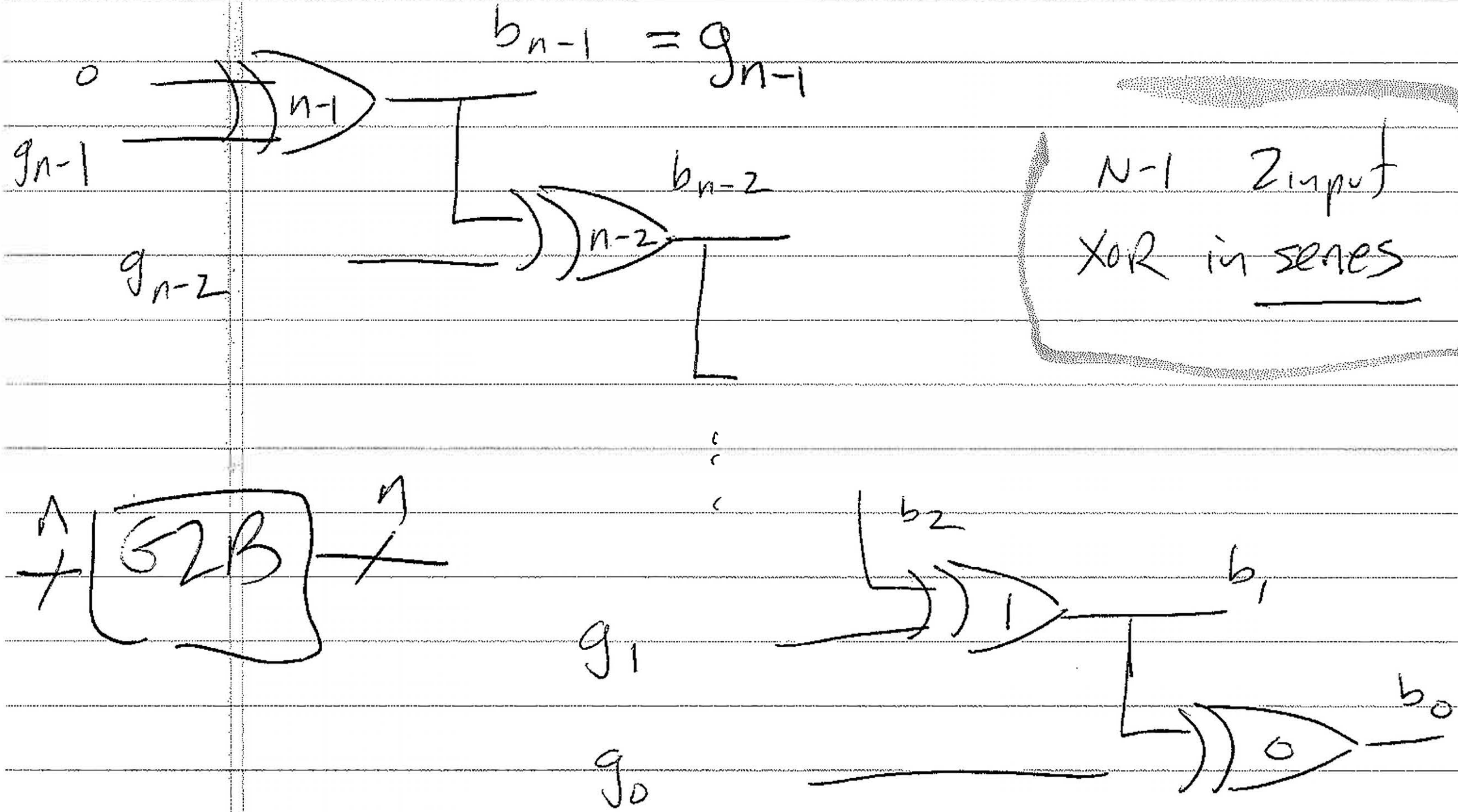
Binary to Gray (B2G)





$g_{n-1} = b_{n-2}$

N-1
2-input
XOR in parallel



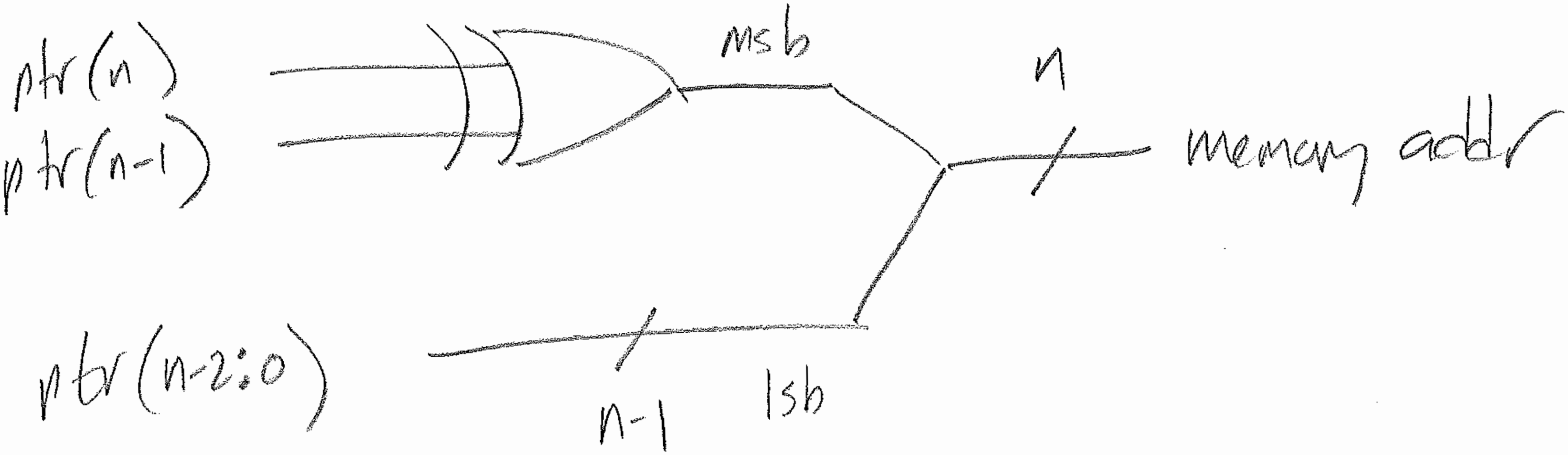
N-1 2input XOR in series

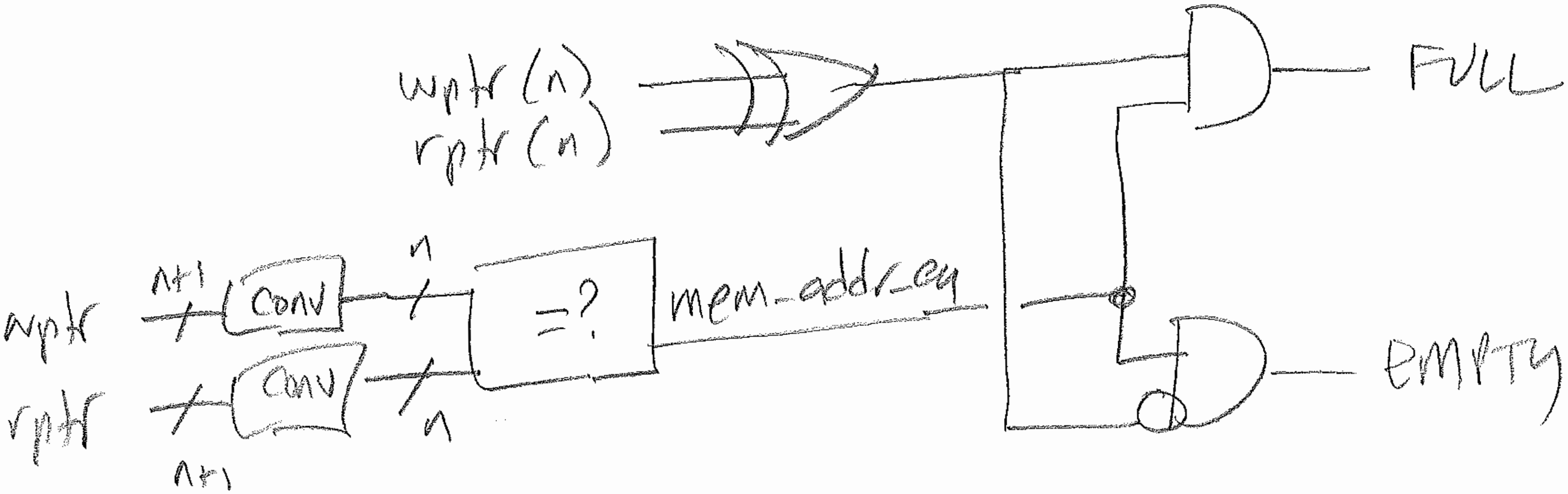
Rather than series for gate 0 could use a parity tree

Full or Empty?!?!

- As with binary pointers, it was suggested to use an augmented Gray code pointer, with one extra bit than needed to address the memory
- However, unlike binary pointers, we can’t simply compare the msb
- Instead, the approach presented in class is to first convert the augmented pointer to the memory address
- If the two sides are accessing the same memory address, then the msb of the augmented pointers can be used to determine Full or Empty

Gray Pointer to Memory Addr (n-bit)





wptr(n)
rptr(n)
FULL
wptr
n+1
conv
n
=?
mem_addr_eq
rptr
conv
n
n+1
EMPTY

"Augmented" (n:0) (n+1 bit)		Mem Addr (n-1:0) (n-bit)
4-bit Gray		3-bit Gray
WP → 0000 ← RP		000
0001		001
0011		011
0010		010
0110		110
0111		111
0101		101
0100		100
1100		000
1101		001
1111		011
1110		010
1010		110
1011		111
1001		101
1000		100

Empty

Do need a mechanism of resetting both interfaces.

	4-bit Gray		3-bit Gray
	0000		000
	0001		001
	0011	← RP	011
	0010		010
	0110		110
WP →	0111		111
	0101		101
	0100		100
	1100		000
	1101		001
	1111		011
	1110		010
	1010		110
	1011		111
	1001		101
	1000		100

5 writes followed by 2 reads,

3 spaces still "valid" (unread)

Remember: WP points to the location to be filled upon a write (the incr)

RP points to location to be read, <u>afterwhich</u> it is incremented.

4-bit Gray		3-bit Gray
0000		000
0001		001
0011	← RP	011
0010		010
0110		110
0111		111
0101		101
0100		100
1100		000
1101		001
WP → 1111		011
1110		010
1010		110
1011		111
1001		101
1000		100

After 5 more

FIFO Full

8 valid (unread) locations

the RP and WP point to the same memory location