CS315-01 Components and Sequential Logic

max2

a_1	a_0	b_1	b_0	r_1	r_0	ri_1
0	0	0	0	0	0	1
0	0	0	1	0	1	1
0	0	1	0	1	0	0
0	0	1	1	1	1	0
0	1	0	0	0	1	1
0	1	0	1	0	1	1
0	1	1	0	1	0	0
0	1	1	1	1	1	0
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	1	0	0
1	0	1	1	1	1	0
1	1	0	0	1	1	0
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	1	1	0

$ri_1 = (\bar{a}_1 \cdot \bar{a}_0 \cdot \bar{b}_1 \cdot \bar{b}_0) + (\bar{a}_1 \cdot \bar{a}_0 \cdot \bar{b}_1 \cdot b_0) + (\bar{a}_1 \cdot a_0 \cdot \bar{b}_1 \cdot \bar{b}_0) + (\bar{a}_1 \cdot a_0 \cdot \bar{b}_1 \cdot b_0)$

$r_1 = \overline{ri_1}$

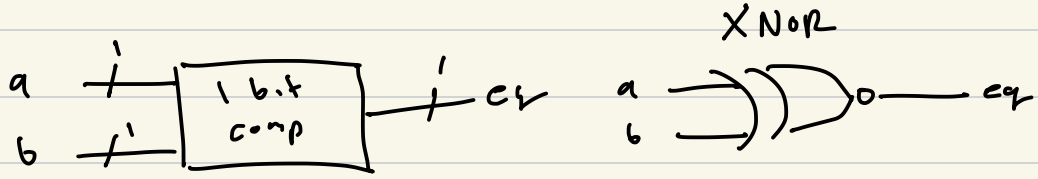
Project 05

Combinational Logic

Comparator is a == b ?

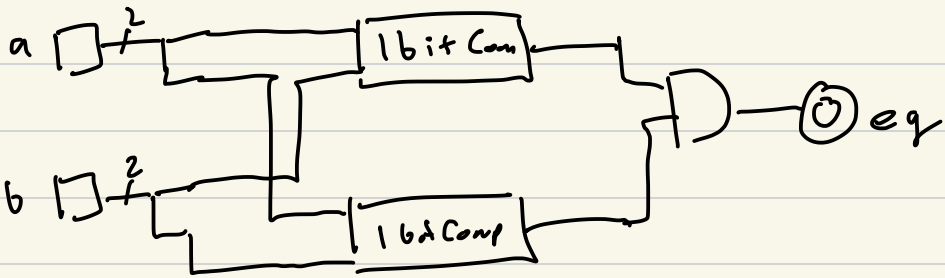
a	b	eq?	XOR	XNOR
0	0	1	0	1
0	1	0	1	0
1	0	0	1	0
1	1	1	0	1

1 bit comparator



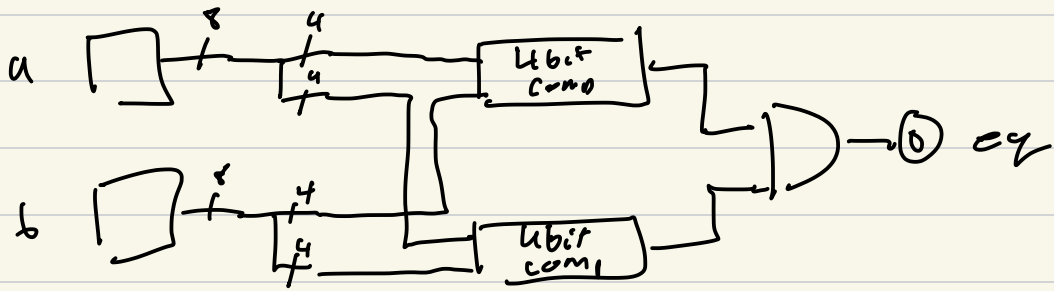
if (a == b)

2 - bit comparator



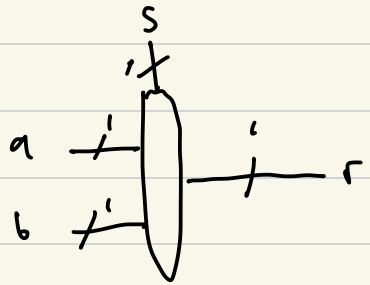
N - bit comparator

8 - bit

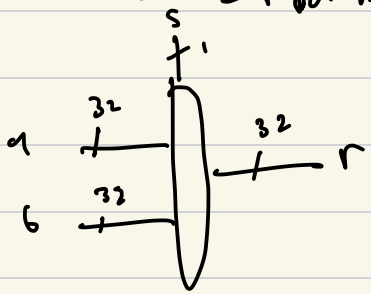


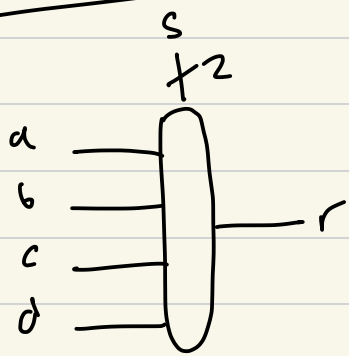
Multiplexor (MUX)

1 bit 2 input MUX

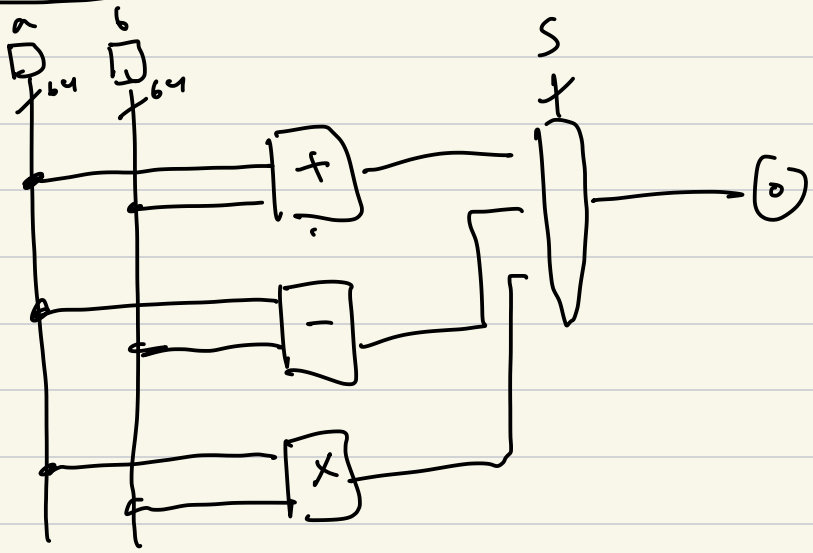


32 bit 2 input MUX





ALU



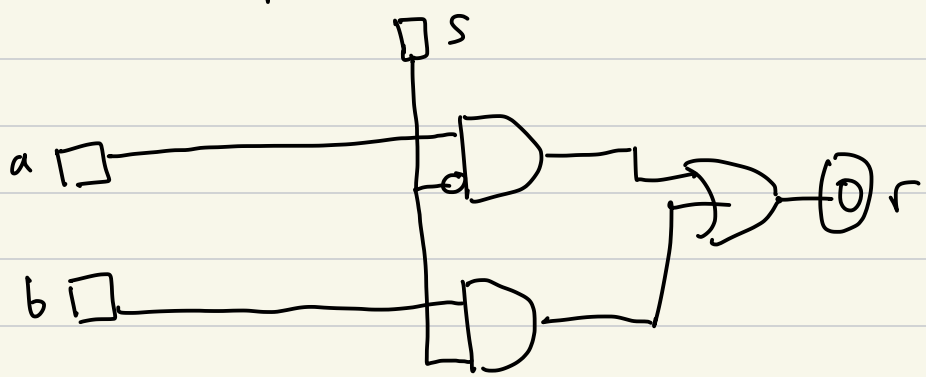
1 bit 2 input MUX

a	b	s	r
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

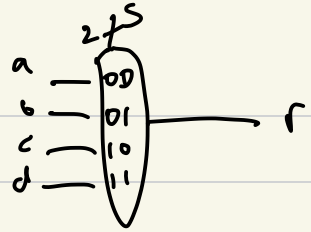
Sum-of-products

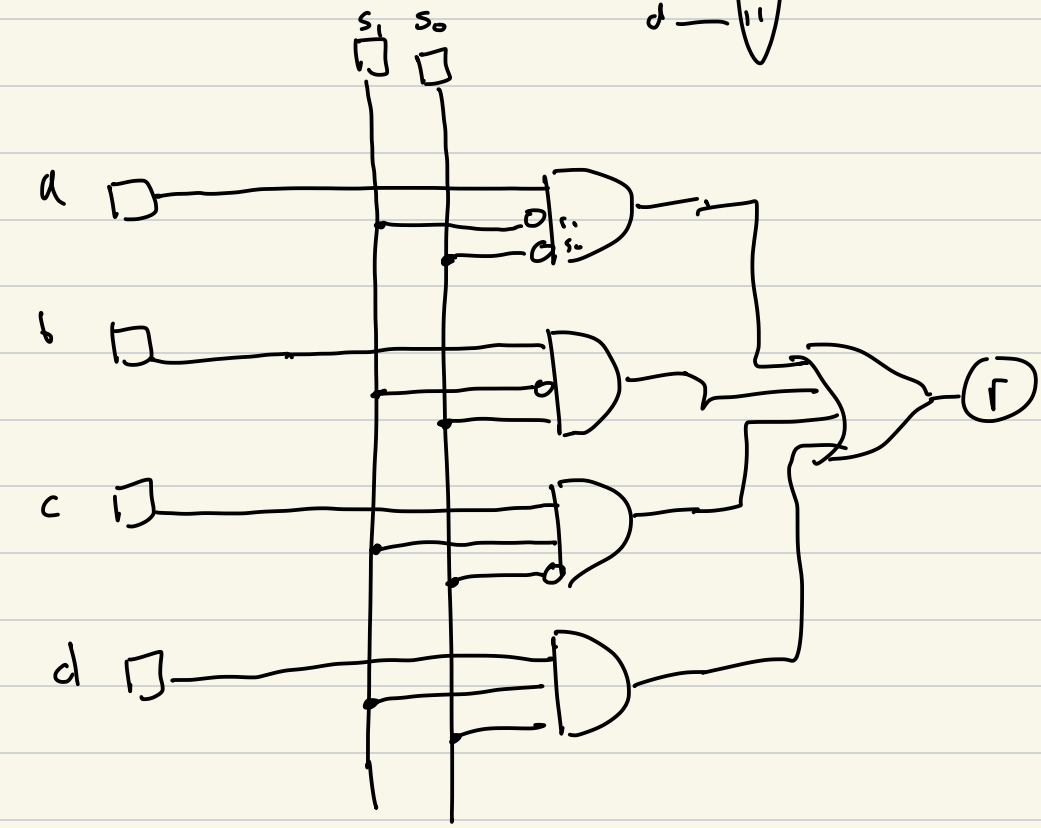
r = ( ) + ( ) + ...

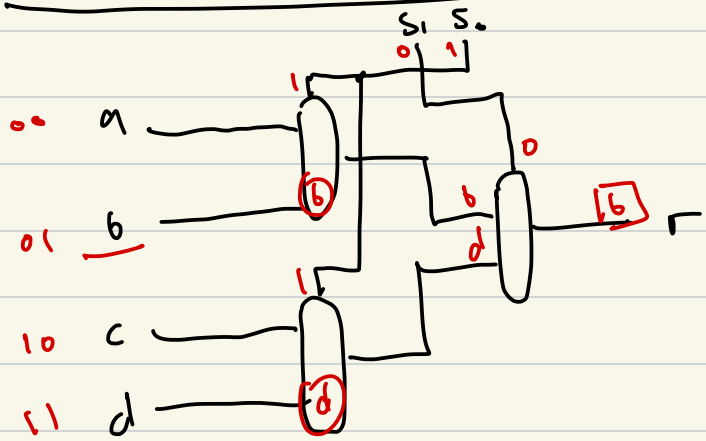
Direct implementation



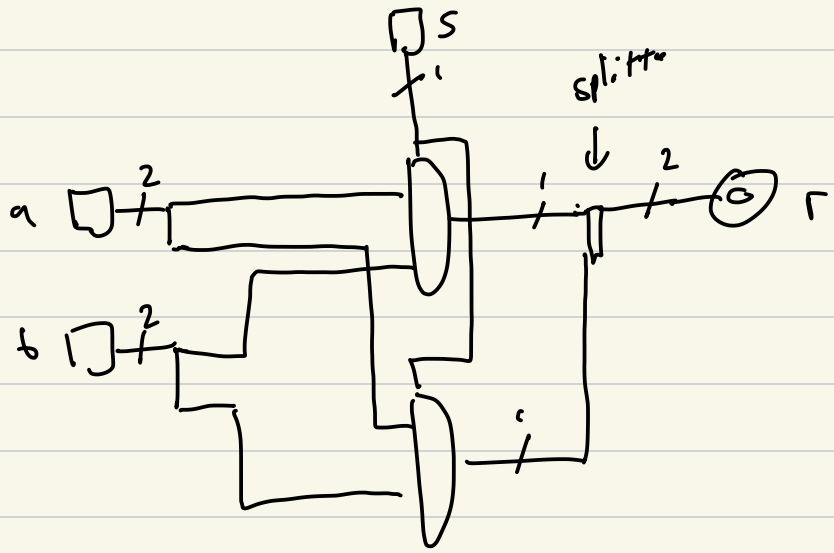
1 bit 4 input MUX



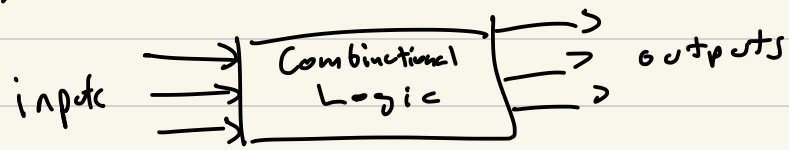


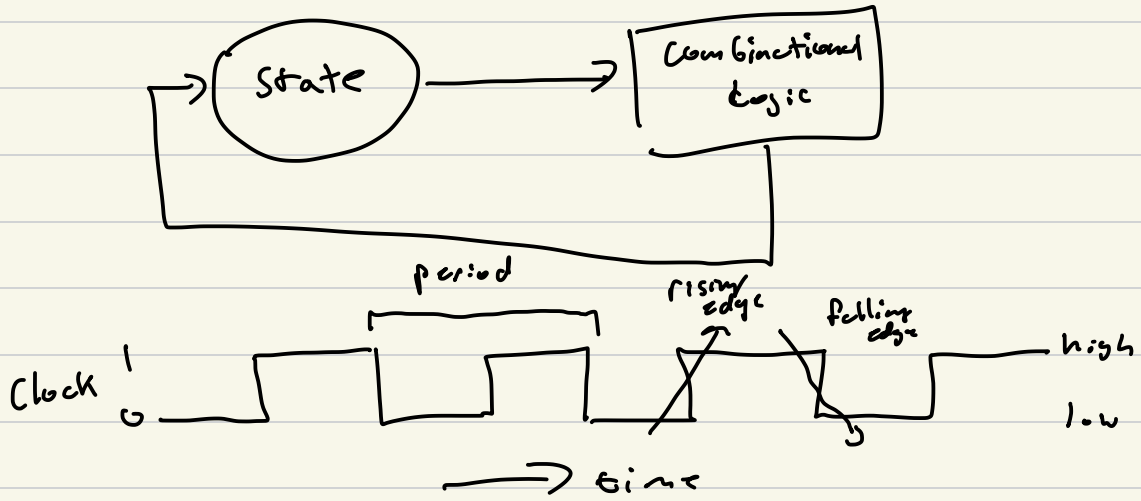


2 bit 2 input MUX



Sequential Logic



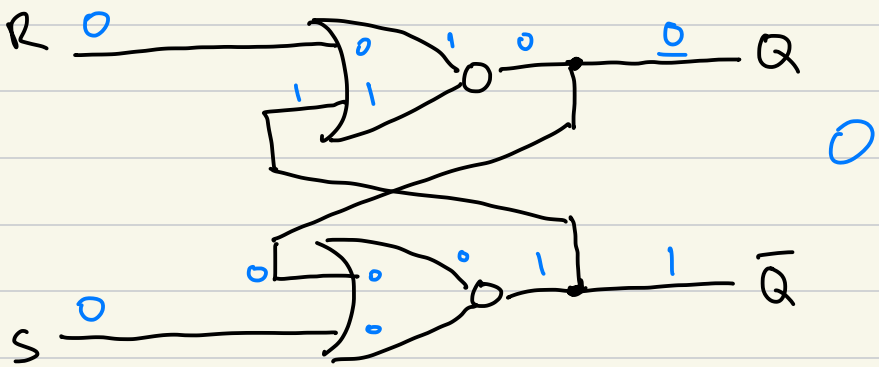


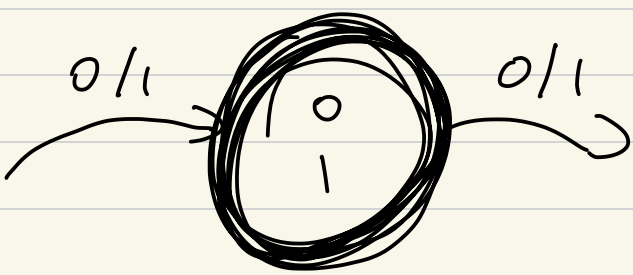
DAG

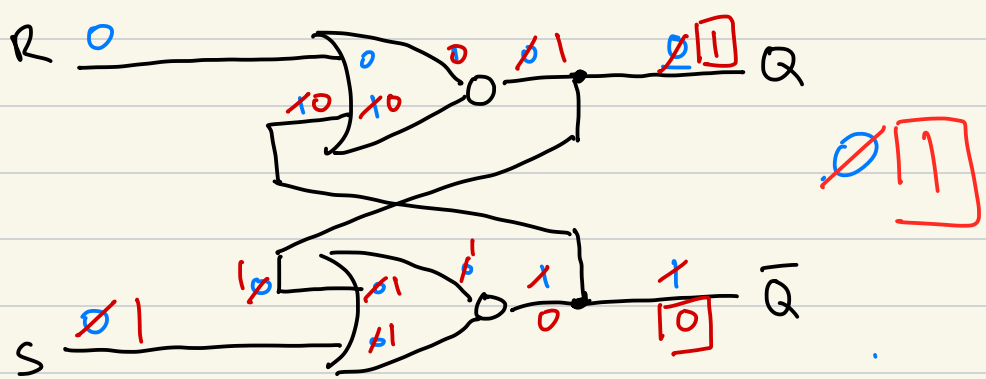
How to store a 1 bit value?

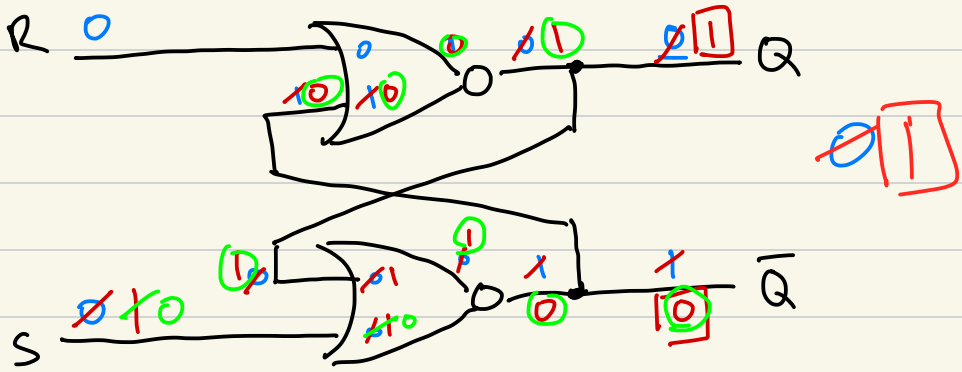
SR Latch set / reset

Static RAM









Eimr

R	S	Q	$\bar{Q}$	
1	0	0	1	
0	0	0	1	
0	1	1	0	
0	0	1	0	
1	1	X	X	undefined