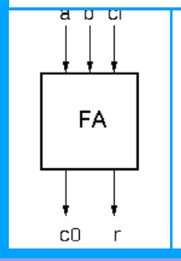
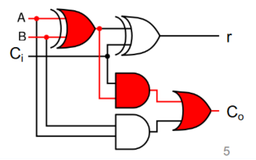
Full Adder

R= Cin xor (A xor B)
Cout = Cin(A xor B) + AB



Critical path:
The longest path from an input to an output



5

Ripple-Carry Adder

$r_i = a_i$ XOR b_i XOR c_{in}

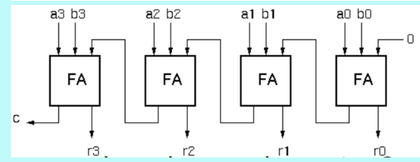
$c_{out} = c_{in}(a_i$ XOR $b_i) + a_i b_i$

Critical path Gates

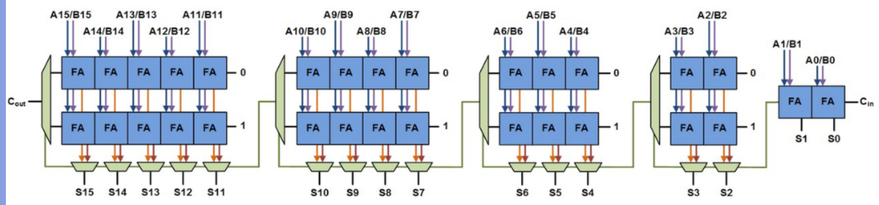
C_{n+1} = xor + 2n(and/or)

S_n = 2xor + 2(n-1)(and/or)

n = bits



Carry Select Adder



From right to left increase size of each block to better match delays

16-bit adder Carry select adder with variable block sizes [5 4 3 2 2]

$T_{total} = (2*sqrt(N)-1)\ T_{FA}$

– assuming $T_{FA} = T_{MUX}$

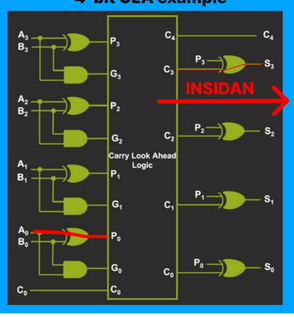
CSACritical Path=
2xor+2(n−1)(and/or)+2(for MUX)

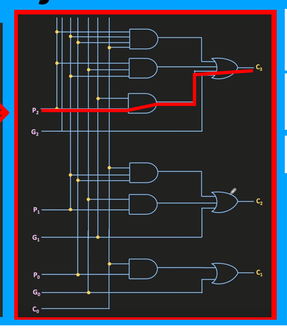
Warning, critical path might be wrong!

For ripple adder $T_{total} = N\ T_{FA}$

Carry Look-ahead adder

4-bit CLA example





Critical path=
log2(n)*(AND/OR) + 2*XOR

ex. 4 bit critical path (S3)=
1xor + 2 (and/or) + 1 xor

$c_{i+1} = g_i + p_i c_i$ $S_i = p_i \oplus c_i$

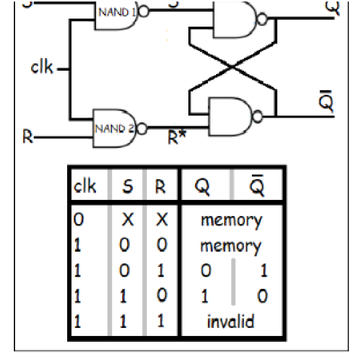
$G_n = A_n B_n$ $P_n = A_n \oplus B_n$

$C_1 = G_0 + P_0 C_0$

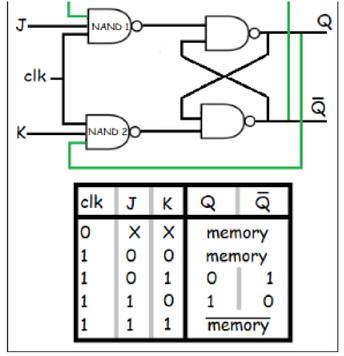
$C_2 = G_1 + P_1 C_1 = G_1 + P_1 G_0 + P_1 P_0 C_0$

$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$

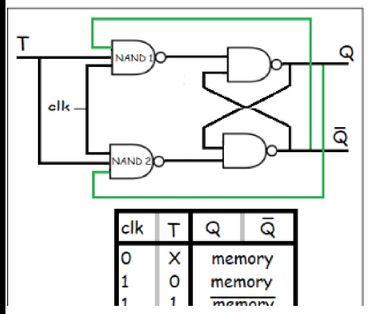
$C_4 = G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$



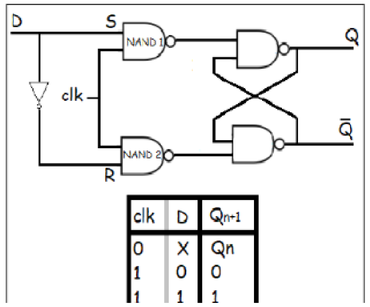
clk	S	R	Q	Q̄
0	X	X	memory	
1	0	0	memory	
1	0	1	0	1
1	1	0	1	0
1	1	1	invalid	



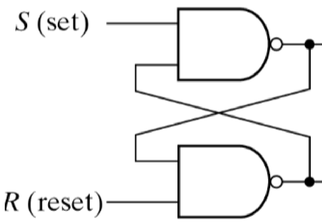
clk	J	K	Q	Q̄
0	X	X	memory	
1	0	0	memory	
1	0	1	0	1
1	1	0	1	0
1	1	1	memory	



clk	T	Q	Q̄
0	X	memory	
1	0	memory	
1	1	memory	

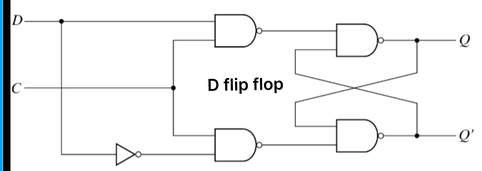


clk	D	Qn+1
0	X	Qn
1	0	0
1	1	1



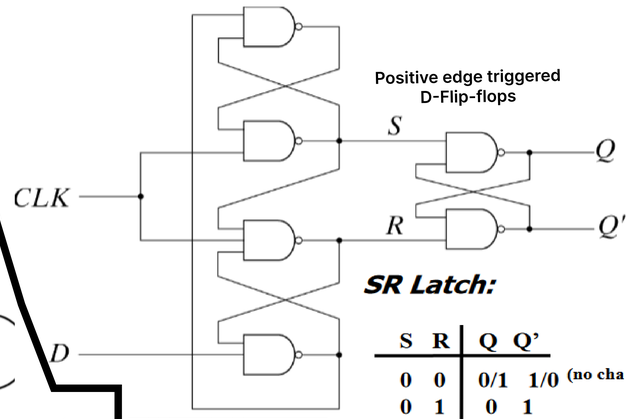
NAND-gate based SR latch

S	R	Q	Q'
1	0	0	1
1	1	0	1
0	1	1	0
1	1	1	0
0	0	1	1



D flip flop

C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

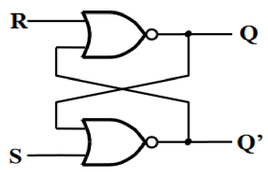


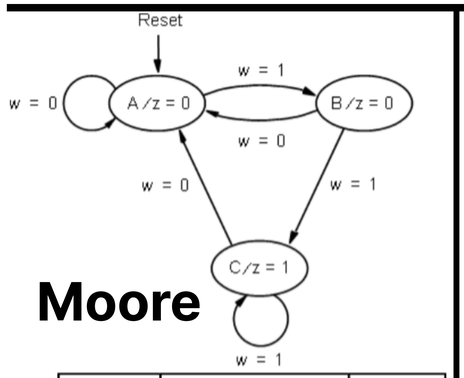
Positive edge triggered D-Flip-flops

SR Latch:

S	R	Q	Q'
0	0	0/1	1/0 (no cha
0	1	0	1
1	0	1	0
1	1	0	0

(b) Truth table

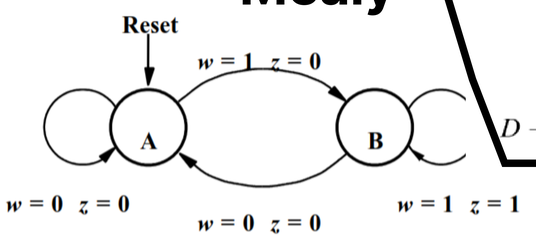




Moore

	Present state	Next state w = 0	Next state w = 1	Output
	$y_2 y_1$	$Y_2 Y_1$	$Y_2 Y_1$	z
A	00	00	01	0
B	01	00	10	0
C	10	00	10	1
	11	*dd*	*dd*	*d*

Mealy



	Present state	Next state w = 0	Next state w = 1	Output w = 0	Output w = 1
	y	Y	Y	z	z
A	0	0	1	0	0
B	1	0	1	0	1

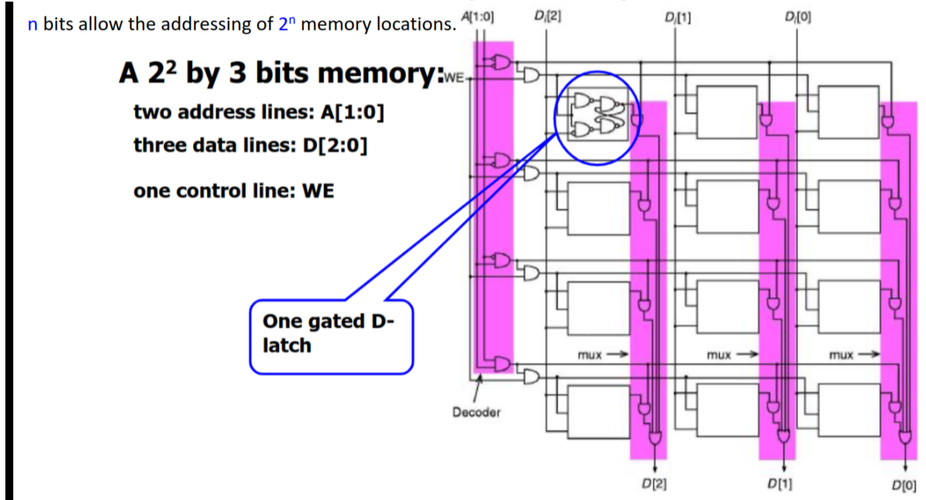
n bits allow the addressing of 2^n memory locations.

A 2^2 by 3 bits memory:

two address lines: A[1:0]

three data lines: D[2:0]

one control line: WE



Use memory blocks that have $2^{[x]}$ entries and each entry has [y]Bytes, to construct a memory of which has $2^{[x+z]}$ entries and [j]*[y] Bytes per entry.
Draw the block diagram of the memory and show how the address bits are connected.

Column is total amount of bits: 16bits x 3 columns = 48 bits
Row is total amount of entries: 512 entries x 8 = 4096 → 4K
Needs 9 bits A8-A0 to address if block has 512 (2^9) entries

Each bit
- is a gated D-latch

Each location
- consists of w bits (here w = 1)
- w = 8 if the memory is byte addressable

Addressing
- n locations means **$\log_2 n$** address bits (here 2 bits => 4 locations)
- decoder circuit translates address into 1 of n locations

giga	G	1 000 000 000	10^9
mega	M	1 000 000	10^6
kilo	k	1 000	10^3
hekto	h	100	10^2

format	Number	r	Integer	Value	
1.3	1.011	0.125 (1/8)	11	1.375	(11/8)
s1.3	01.011	0.125 (1/8)	11	1.375	(11/8)
s1.3	11.011	0.125 (1/8)	-5	-0.625	(-5/8)
2.4	10.0111	0.0625 (1/16)	39	2.4375	(39/16)

Suppose you need to represent distance from 1 millimeter (mm) to 1 meter (m) with an accuracy of 1%.

1% of a mm accuracy is 10 micrometers (μm).
This needs a resolution of 2*10 μm = 20 μm = $2*10^{-5}$ meters = 1/50000 m.
If meters is the integer part we need, then we need 1 bit for integer (to count between 0-1 meters). Then, the fraction needs 16 bits to count $1/2^{16}$ = 1/64*1024) meters (which is a bit shorter than the 1/50000 meters resolution). So the fixed-point representation would require 17 bits (1.16).

Alternatively, we can count multiples of 20 μm which is the required resolution. 1 meter = 50000 * 20 μm for which we need again 16 bits (2^{16}=64K).

For a floating-point representation, the mantissa needs to be only 6 bits to offer accuracy of 1%. So, the resolution needs be double that: 2% = 1/50; 1/64= $1/2^6$ should then be enough (6 bits mantissa). The dynamic range is from 1mm to 1 m (10^3 mm) so it is 10^3, which is smaller than 1024 = 2^{10}. Then, 4 bits are enough to count up to 10, so with 4-bits exponent we can represent a range of 2^{16} (which is larger than 2^{10}). With 4 bits exponent the maximum value of the exponent part is 2^{15} which is larger than the maximum value we need considering the unit is mm (10^3). The closest power of two larger than 10^3 is 2^{10}. So, we can set the bias to be 15-10 =5 so the maximum exponent to be $2^{15-5} = 2^{10}$ =1024 >1000, since the largest number to represent is 1000 mm. So, the floating point format is 6 bits mantissa and 4 bits exponent.

Question: 1s to 10^4s \w 5% accuracy:
FIXED 1. Acc = 0,05s
Resolution = 2*Acc = 2*0,05 = 0,1s 2. 2^4 16 > 10 thus 4 bits to represent the fraction part.
3. 1 to 10^4 (10 000), 2^14 is ~16k > 10k thus 14 bits to represent integer part.
ANS: 18 bits in total (14.4)
FIXED ALT: We can count 1/10 seconds which is the required resolution. 10^4 seconds = 10^5 tenths of seconds for which we need 17 bits (2^17 = 128*1024 = 131072).
FLOAT 1. Acc = 0,05s
Resolution = 2*Acc = 2*0,05 0,1s 2. 2^4 = 16, 16 > 10 thus Mantisa = 4 bits.
3. 1-10^4 (10 000), 2^14> 10^4, to store num 14 we need 4 bits.
4. ((2^4) - 1) = 15 thus we need bias = 1.
ANS: Mantisa = 4bits, Exponent = 4bits, Bias = 1. Total: 8 bits
C. ANS: Floating point needs fewer bits than fixed point, so it is a better choice. That happens because the required accuracy is a percentage and not a fixed absolute value.