03 Logic networks

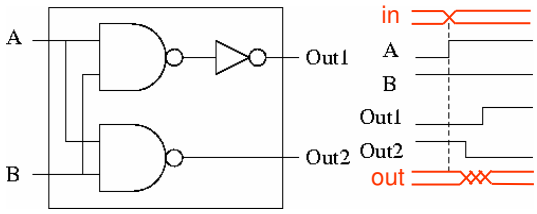
03.04 Gate-level design

- Design metrics
- Design styles
- Examples
- Adders

Design metrics

- Area (**A**)
  - Number of gates
  - Number of 2-input NANDs
  - Number of gates inputs
- Performance
  - *Propagation* time (delay): pin-to-pin, overall (**Tp**)
  - *Contamination* time: pin-to-pin, overall (**Tc**)
  - *Throughput* (**rate**)
- Power
  - Static (**W**)
  - Dynamic (**W**)

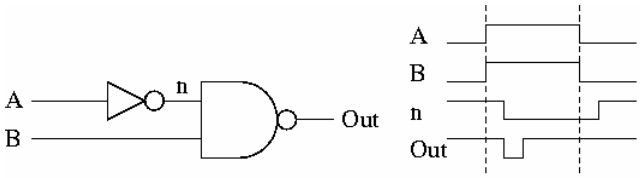
Prop. and Cont. Time (ex1)



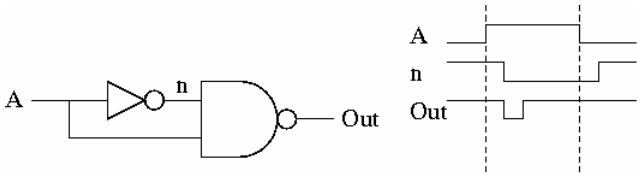
Tp	Out1	Out2
A	2	1
B	2	1

Tc	Out1	Out2
A	2	1
B	2	1

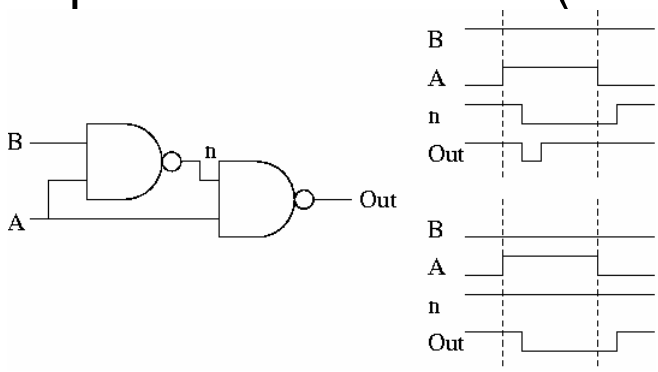
Prop. and Cont. Time (ex2)



Prop. and Cont. Time (ex3)



Prop. and Cont. Time (ex4)

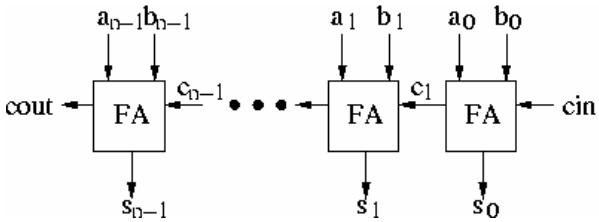


Design approaches

- *Logic synthesis*:
  - General
  - Inefficient
  - Non-scalable
  - Example: Boolean functions of a few variables
- *Top-down problem partitioning:*
  - Application-specific
  - Modular
  - Scalable
  - Example: Arithmetic operators

T-D Example: Ripple-carry adder

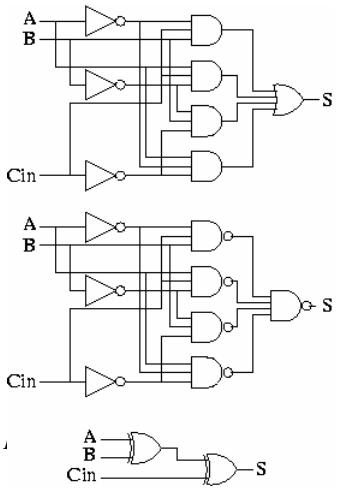
- Functional specification: S=A+B



T-D Example: Full adder (1)

- Functional specification:

Cin	A	B	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

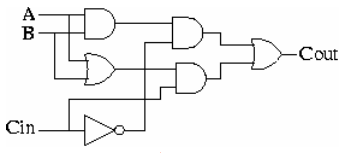


$$S = Cin'A'B + Cin'AB' + CinA'B' + CinAB$$
$$S = Cin'(A'B + AB') + Cin(A'B' + AB)$$
$$S = Cin'(A \oplus B) + Cin(A \oplus B)' = Cin \oplus A \oplus$$

T-D Example: Full adder (2)

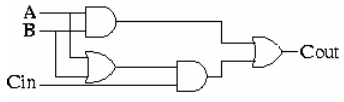
- Functional specification:

Cin	A	B	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



$$Cout = Cin'AB + CinA'B + CinAB' + CinAB$$
$$Cout = Cin'AB + Cin(A + B)$$

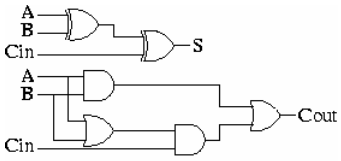
$$Cout = Cin'AB + Cin(A + B + AB)$$
$$Cout = AB + Cin(A + B)$$



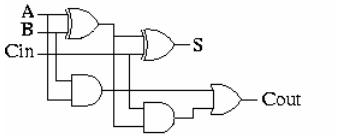
T-D Example: Full adder (3)

- Putting it all together:

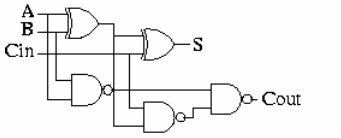
$$S = Cin \oplus (A \oplus B)$$
$$Cout = AB + Cin(A + B)$$



$$Cout = AB + Cin(A \oplus B + AB)$$
$$Cout = AB + Cin(A \oplus B)$$



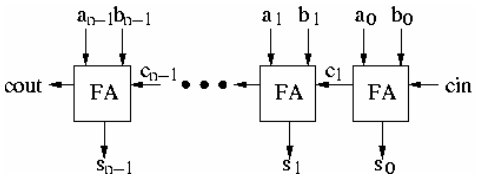
$$Cout = ((AB)'+(Cin(A \oplus B))')'$$



Adders

1. Ripple-Carry Adder (RCA)
2. Synchronous RCA
3. Pipelined RCA
4. Bit-serial Adder
5. Carry-Lookahead Adder

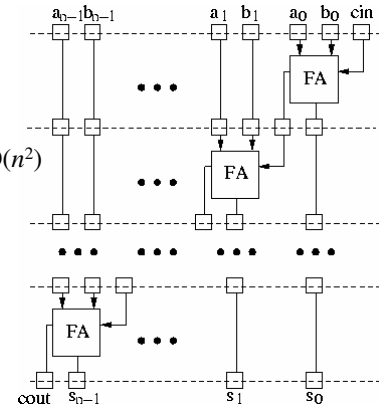
Ripple-carry adder (RCAn)



A(RCAn) = n A(FA) = O(n)
Tp(RCAn) = n Tp(FA) = O(n)
Tc(RCAn) = Tc(FA) = O(1)
Rate(RCAn) < 1/Tp(RCAn) = O(1/n)

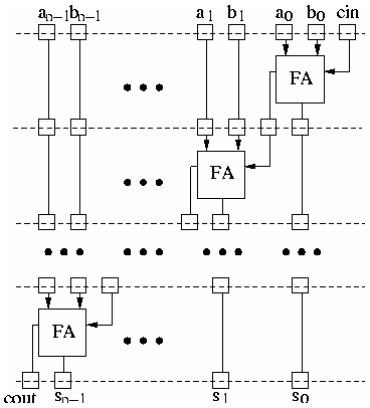
Synchronous RCAn (SRCAn)

A(SincRCAn) = nA(FA) + 2n(n-1)A(FF) = O(n^2)
Tp(SincRCAn) = nTclk > nTp(FA) = O(n)
Tc(SincRCAn) = nTclk > nTp(FA) = O(n)
Rate(SincRCAn) = 1/(nTclk) = O(1/n)



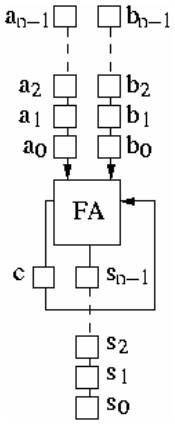
Pipelined RCAn (PRCAn)

A(PRCAn) = nA(FA) + 2n(n-1)A(FF) = O(n^2)
Tp(PRCAn) = nTclk > nTp(FA) = O(n)
Tc(PRCAn) = nTclk > nTp(FA) = O(n)
Rate(PRCAn) = 1/Tclk = O(1)



Bit-serial adder (BSAn)

A(BSAn) = A(FA) + A(FF) = O(1)
Tp(BSAn) = nTclk > nTp(FA) = O(n)
Tc(BSAn) = Tclk > Tp(FA) = O(1)
Rate(BSAn) = 1/(nTclk) = O(1/n)



Carry Lookahead Adder (CLAn)

Observations:

$c_i = a_i*b_i + (a_i+b_i)c_{i-1} = g_i + p_i * c_{i-1}$ (1)

The first term generates the carry out (*generate* $g_i = a_i*b_i$)

The second term propagates the carry (*propagate* $p_i = a_i+b_i$)

Implementation:

$c_i = g_i + p_i (g_{i-1}+p_{i-1} (g_{i-2}+p_{i-2}(\ldots (g_0+p_0*Cin)\ldots)))$ (2)

$c_i = g_i + p_ig_{i-1}+p_ip_{i-1}g_{i-2}+ p_ip_{i-1}p_{i-2}g_{i-3} + \ldots + p_ip_{i-1}p_{i-2} \ldots p_0Cin$ (3)

Carry Lookahead Adder (CLAn)

Unit delay model

$A(CLAn) = A(FA_0)+\ldots+A(FA_{n-1}) = A(FA_0)+\ldots+O(n^2) = O(n^3)$

$Tp(CLAn) = Tp(FA) = O(1)$

$Tc(CLAn) = Tc(FA_0) = O(1)$

$Rate(CLAn) > 1/Tp(CLAn) = O(1)$

Gate delay proportional to the number of inputs

$A(CLAn) = A(FA_0)+\ldots+A(FA_{n-1}) = A(FA_0)+\ldots+O(n^2) = O(n^3)$

$Tp(CLAn) = Tp(FA_{n-1}) = O(n)$

$Tc(CLAn) = Tc(FA_0) = O(1)$

$Rate(CLAn) > 1/Tp(CLAn) = O(1/n)$

Actual

$O(n) < A(CLAn) < O(n^3)$

$O(1) < Tp(CLAn) < O(n)$

$Tc(CLAn) = O(1)$

$O(1/n) < Rate(CLAn) < O(1)$