⊕ Xor operator

AND: | a b | | OR: | a b | | NOT | a | | NAND: | a b | | NOR: | a b |

XOR: | a b | | XNOR: | a b |

AND: OR: NOT: XOR

ADD NOT to OR to get NOR

ADD NOT to XOR to get XNOR

XNOR (Replace NAND with NOR GATE)

Verilog: f=1 when odd # of 1's

When #1's are even

AND: & NOT: ~ XNOR: ~^ or ^~

OR: | XOR: ^

AXIOM:

1. 0·0=0 1+1=1
2. 1·1=1 0+0=0
3. 0·1=1·0=0 0+1=1+0=0
4. if x=0, then $\bar{x}$=1 ; if x=1, then $\bar{x}$=0

DUALITY

5. x·0=0 x+1=1
6. x·1=x x+0=x
7. x·x=x x+x=x
8. x·$\bar{x}$=0 x+$\bar{x}$=1
9. $\bar{\bar{x}}$=x

Commutative

10. x·y=y·x x+y=y+x

ASSOCIATIVE

11. x(yz)=(xy)z x+(y+z)=(x+y)+z

Distributive

12. x(y+z)=(xy)+(xz) x+(yz)=(x+y)(x+z)

ABSORBTION

13. x+xy=x x·(x+y)=x

COMBINING

14. xy+x$\bar{y}$=x (x+y)·(x+$\bar{y}$)=x

DEMORGAN'S

15. $\overline{x \cdot y} = \bar{x} + \bar{y}$ $\overline{x + y} = \bar{x} \cdot \bar{y}$

COVERING

16. x+$\bar{x}$y=x+y x·($\bar{x}$+y)=x·y

CONCENCUS

17. xy+y$\bar{z}$+$\bar{x}$z = xy+$\bar{x}$z (x+y)(y+$\bar{z}$)($\bar{x}$+z) = (x+y)($\bar{x}$+z)

R-S LATCH:

S	R	Q	$\bar{Q}$
0	0	X	X
0	1	0	1
1	0	1	0
1	1	Race cond.	

GATED R-S LATCH:

CLK	S	R	Q(t+1)
0	X	X	Q(t) - storing
1	0	0	Q(t) - oth value
1	0	1	0
1	1	0	1
1	1	1	Race cond.

Sop:

$f(x,y,z) = \bar{x}\bar{y}\bar{z} + \bar{x}y\bar{z} + x\bar{y}\bar{z}$ } Canonical Sop

↳ Include all minterms

Pos:

$f(x,y,z) = (x+y+\bar{z})(x+\bar{y}+z)(\bar{x}+y+z)$ } Canonical Pos

↳ Includes all maxterms

$\bar{f} = (x\bar{y}\bar{z})(x\bar{y}\bar{z})$, $f = \bar{\bar{f}} = (\overline{x\bar{y}\bar{z}})(\overline{x\bar{y}\bar{z}})$

$= (\bar{x}+y+z)(\bar{x}+y+\bar{z})$

K-maps:

Implicant: Any product term which the function equals 1

Literal: A variable or it's component: $x_1, \bar{x}_2, x_3$

Prime Implicant: Make the biggest group with its neighbours

Cover: Any set of implicants that cover all the 1's in the K-map

Essential Prime Implicant: A PI that must be included to form cover (Not repeated by other Groupings)

Minimal Cover: All essential PI + other PI necessary to form a cover.

x_1x_2 \ x_3x_4	00	01	11	10
00			1	1
01			1	1
10	1			
11	1			

$X_2\bar{X}_3$

↳ Keep the one that is constant

- Groups in 2, 4, 8, 16...

* Simplify:

ADD A second or third term to simplify:

$xy + y\bar{z} + \bar{x}\bar{z} \rightarrow xy(z+\bar{z}) + y\bar{z}(x+\bar{x})$..

$x+\bar{x} = 1$ ↑

COST:

# of Gates + # of inputs

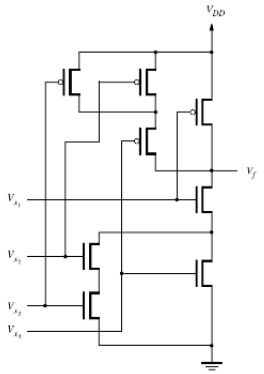
Example B.2 Consider the function

$$f = \bar{x}_1 + (\bar{x}_2 + \bar{x}_3)\bar{x}_4$$

Then

$$\bar{f} = x_1(x_2x_3 + x_4)$$

These expressions lead directly to the circuit in Figure B.17.



GATED D-LATCH

CLK	D	Q(t+1)
0	X	Q(t) - old value
1	0	0
1	1	1

Latch is level sensitive
FF is edge sensitive

FULL ADDER:

$S_i = C_{in} \wedge a \wedge b$

$C_{out} = (b\&a) | (C_{in}\&a) | (C_{in}\&b)$

HALF ADDER

CMOS Circuits:

PMOS	NMOS
Closed when high	Closed when low
open when low	open when high
Strong logic 1	Strong logic 0

$f = \bar{x}_1 + \bar{x}_2\bar{x}_3$

Draw PUN first, then draw complementary PDN after

OR - Parallel

AND - Series

Parallel P = Series N

Series P = Parallel N

Inverter

D Flip Flop (+'ve edge)

ADD NOT infront of CLK to get -'ve edge

Pos edge neg edge

COUNTER (SYNC) - Count up

COUNTER (SYNC) - Count Down

COUNTER: (Async)

Tri State

Tri-State

Figure B.65 A 2 x 2 array of SRAM cells.

Carry Ripple Adder

Carry-Lookahead adder

Complex CMOS Gates:

$f = \bar{x}y + x\bar{y}$	f Is pull up network
$f = (x+\bar{y})(\bar{x}+y)$	$\bar{f}$ Is pull down network
$\bar{x}y$	Is in series
$x+\bar{y}$	Is in parallel
$(x+\bar{y})(\bar{x}+y)$	Is two parallel circuits in series

Comparison of 16 bit adders to count to 16:
Carry Ripple: 33
Lookahead: 3 (Large fan in)
Ripple Carry + Lookahead: 9 (fan-in ≤5)
Hierarchial Lookahead: 5 (fan-in ≤5)

Table 3.2 Interpretation of four-bit signed integers.

$b_3b_2b_1b_0$	Sign and magnitude	1's complement	2's complement
0111	+7	+7	+7
0110	+6	+6	+6
0101	+5	+5	+5
0100	+4	+4	+4
0011	+3	+3	+3
0010	+2	+2	+2
0001	+1	+1	+1
0000	+0	+0	+0
1000	−0	−7	−8
1001	−1	−6	−7
1010	−2	−5	−6
1011	−3	−4	−5
1100	−4	−3	−4
1101	−5	−2	−3
1110	−6	−1	−2
1111	−7	−0	−1

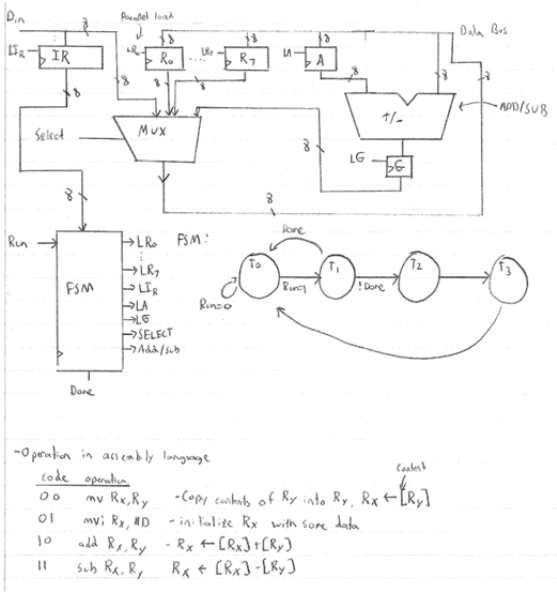
Fast Carry Adders:

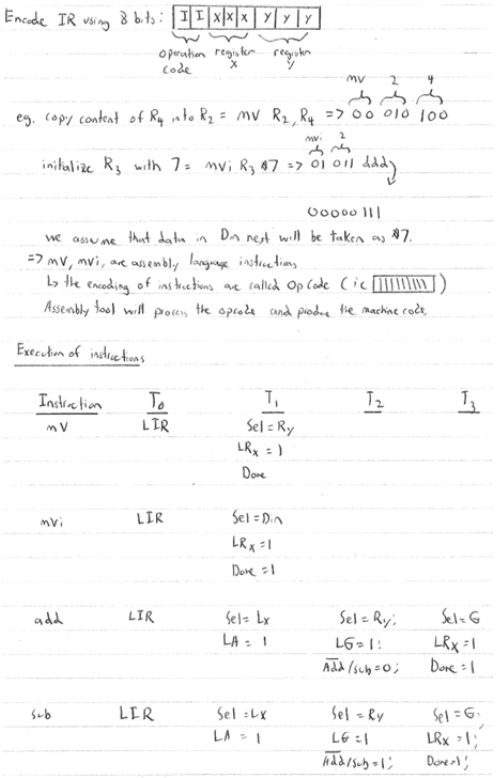
$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$

$= x_i y_i + (x_i + y_i)c_i = g_i + p_i c_i$

$g_i = x_i y_i \quad p_i = x_i + y_i$

Value	Power: 2^n, n=
1	1
2	2
4	3
8	4
16	5
32	6
64	7
128	8
256	9
512	10
1024	11
2048	12

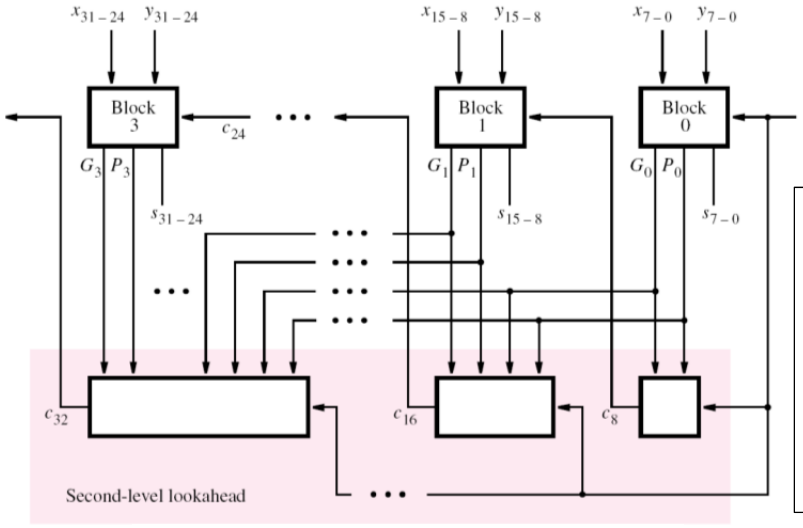




Minimizing State Machines:
Split into First Partition: P1=ABCDEFG
Partition by Output P2 = (ABD)(CEFG)
Make sure that the 1's next state and 0's next state go to the same partition.
Split the First partition into more partitions until they all go to the same partitions.

-Operation in assembly language

code	operation	
00	mv R_x, R_y	-Copy contents of R_y into R_x, $R_x \leftarrow [R_y]$ (Content)
01	mvi R_x, #D	- initialize R_x with some data
10	add R_x, R_y	- $R_x \leftarrow [R_x] + [R_y]$
11	sub R_x, R_y	$R_x \leftarrow [R_x] - [R_y]$



Hierarchial Carry-Lookahead Adder

Two's complement:

1. Invert the numbers
2. Add one

Shannon's Theorem:

$f = \bar{x}\bar{y}z + \bar{x}zw + x\bar{z}\bar{w} + xy\bar{z}$

$f = \bar{x}(\bar{y}z + wz) + x(y\bar{z} + \bar{w}\bar{z})$

$f_{\bar{x}} = z(\bar{y} + w) \qquad f_x = \bar{z}(y + \bar{w})$

x is the control for the multiplexer, as is z

Another method for xf_x and $\bar{x}f_{\bar{x}}$:

f_x = Set all $x = 1, \bar{x} = 0$

$f_{\bar{x}}$ =Set all $x = 0, \bar{x} = 1$

Timing Analysis:

Min clock period: Critical Path timing

$t_{min} = t_{cQ} + t_{AND} + t_{OR} + t_{NOT} + t_{su}$

Hold time for Flip-Flops: Shortest Path timing

-Make sure shortest path ≥ hold time

$f_{max} = \frac{1}{T_{min}}$

Clock Skew: $T_{newmin} + \Delta = T_{min}$

$Max\ \Delta = T_{shortest} - T_{hold}$

t_{su} – Time data must be stable before clock edge

t_h – Time data must be stable after clock edge

t_{cQ}- Time from clock edge to when output Q is stable

t_{cQ} Is for exiting from FF and t_{su} is for entering FF