EN1600
Design and Implementation of VLSI Systems
Fall 2016

Lectures 20, 21: Multiplication, Power Dissipation (revisited)

Reading: Chap. 11, section 11.9 November 21, 23, 2016
Chap. 2, sec. 2.4.3, 2.4.4, Chap. 5, sec. 5.3 Prof. R. Iris Bahar
Weste & Harris

BROWN

BROWN

What's coming up?

- Office hours today: 11am-1pm
- Final Project Proposals due this Wednesday by noon
- Remaining lectures:
  - Finish Adders
  - Multipliers
  - Datapath Logic
  - Design for reliability, low power circuits, emerging topics…
- Final exam: ***Wednesday, November 30 (in class)***
- Final Presentations: Mon., Dec. 13, 9am-noon, B&H 141

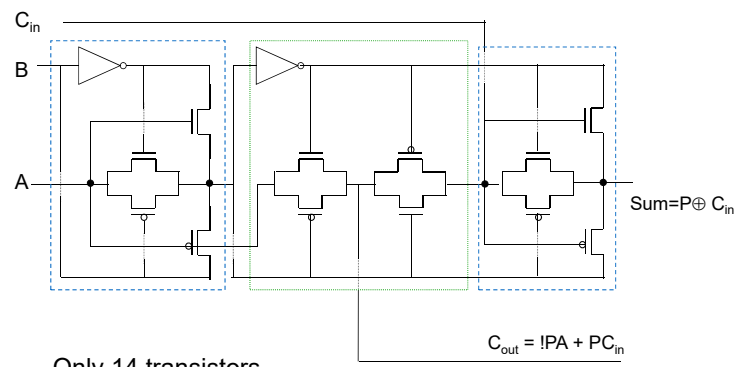
BROWN

Another Transistor level implementation

- Remember Propagate signal $P=A \oplus B$
  - $S=A \oplus B \oplus C_{in}=P \oplus C_{in}$
  - $C_{out}=AB+AC_{in}+BC_{in}=PC_{in}+\bar{P}A$
  - P and S are XOR functions, C_{out} is a mux function
  - Notice that P is shared between S and C_{out}

BROWN

TG full adder



Only 14 transistors
Where is the critical path?

BROWN

Fast Carry Chain Design

- The key to fast addition is a low latency carry network
- What matters is whether in a given position a carry is
  - generated $G_i = A_i \,\&\, B_i = A_iB_i$
  - propagated $P_i = A_i \oplus B_i$ (sometimes use $A_i + B_i$)
  - annihilated (killed) $K_i = !A_i \,\&\, !B_i$
- Giving a carry recurrence of

$$\boxed{C_{i+1} = G_i + P_iC_i}$$

$C_1 = G_0 + P_0C_0$

$C_2 = G_1 + P_1C_1 = G_1 + P_1G_0 + P_1\,P_0\,C_0$

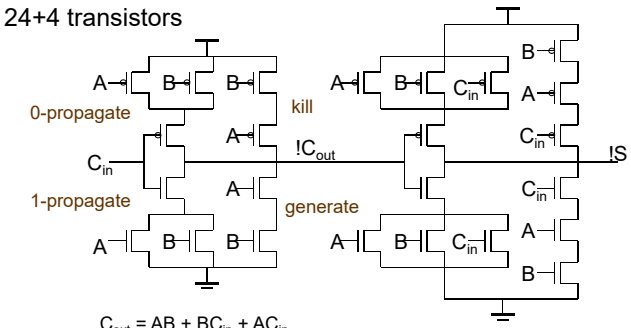
$C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0\,C_0$

$C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0\,C_0$

BROWN

Mirror Adder

24+4 transistors

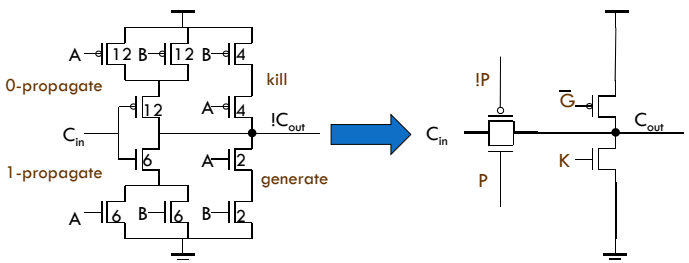


$C_{out} = AB + BC_{in} + AC_{in}$

$SUM = ABC_{in} + !C_{OUT}(A + B + C_{in})$

BROWN

Mirror adder



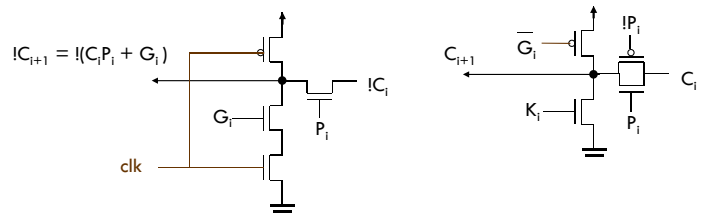
$C_{i+1} = G_i + P_iC_i$

Note that P, K, and G are all mutually exclusive

BROWN

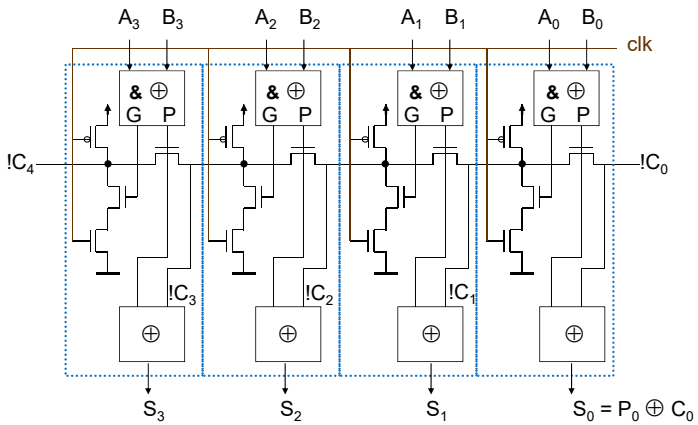
Manchester Carry Chain

- Switches controlled by G_i and P_i

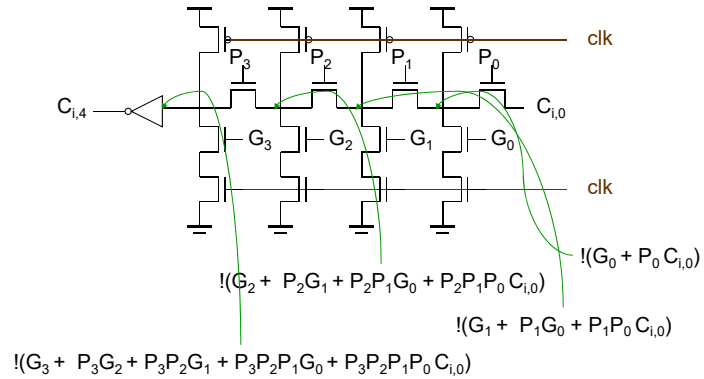


- Total delay of
  - time to form the switch control signals G_i and P_i
  - setup time for the switches
  - signal propagation delay through N switches in the worst case

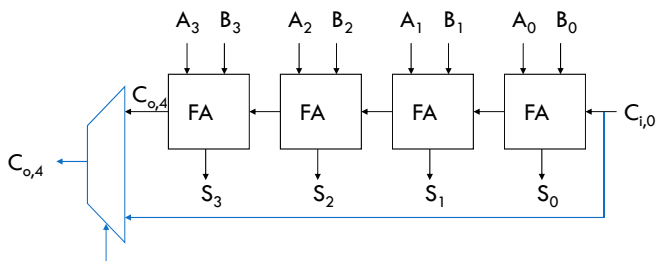
4-bit sliced MCC adder



Domino Manchester carry chain



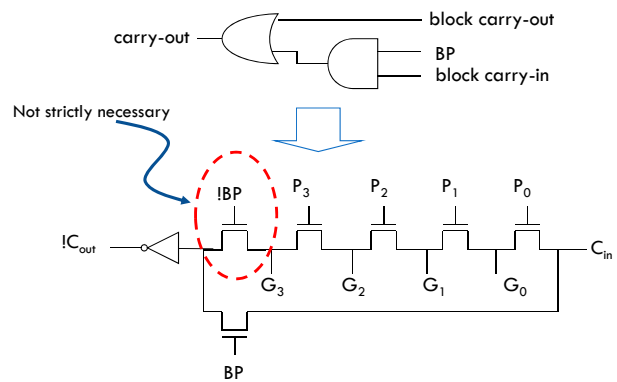
Carry-skip (carry-bypass) adder



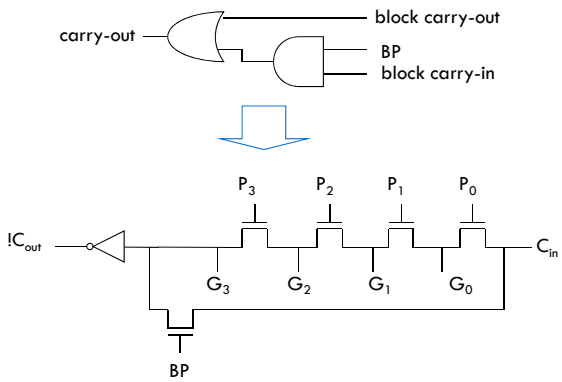
BP = $P_0\ P_1\ P_2\ P_3$ "Block Propagate"

If (P_0 & P_1 & P_2 & P_3 = 1) then $C_{o,4} = C_{i,0}$ otherwise the block itself kills or generates the carry internally (and doesn't need $C_{i,0}$ to compute $C_{o,4}$)

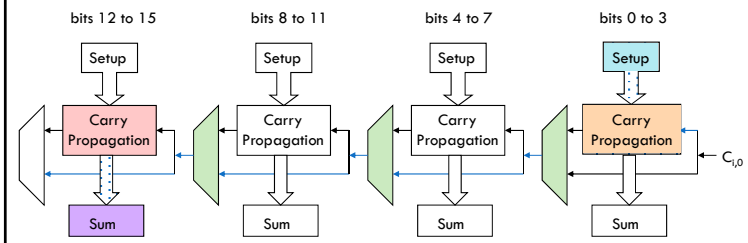
Carry-skip chain implementation



Carry-skip chain implementation



4-bit block carry-skip adder



Worst-case delay → carry from bit 0 to bit 15
carry generated in bit 0, ripples through bits 1, 2, and 3, skips the middle two groups (B is the group size in bits), ripples in the last group from bit 12 to bit 15

$$T_{add} = t_{setup} + B\, t_{carry} + ((N/B) - 1)\, t_{skip} + B\, t_{carry} + t_{sum}$$

Static vs. Variable Block Size

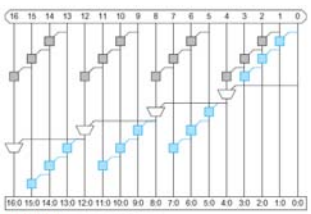


FIGURE 11.18 Carry-skip adder PG network

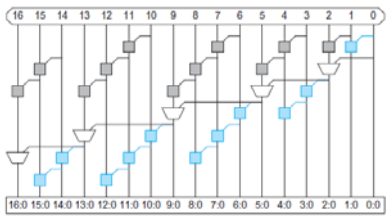
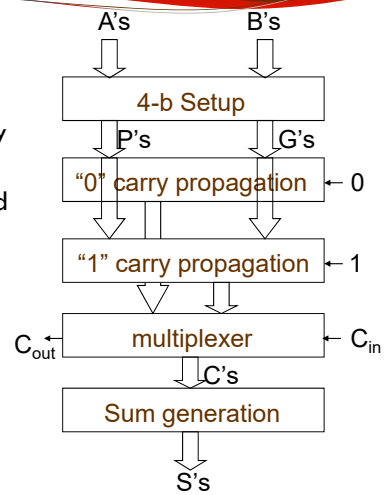


FIGURE 11.19 Variable group size carry-skip adder PG network

Carry select adder

- Pre-compute the carry out of each block for both carry_in = 0 and carry_in = 1 (can be done for all blocks in parallel) and then select the correct one



Carry Select Adder

- AND/OR Mux selects "carry-1" or "carry-0" block depending on carry in of previous stage
- Here, C_4 starts the Mux selection process.
- Compared to carry skip, avoids having to wait for the ripple carry of the last block.

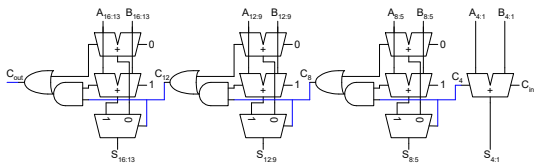
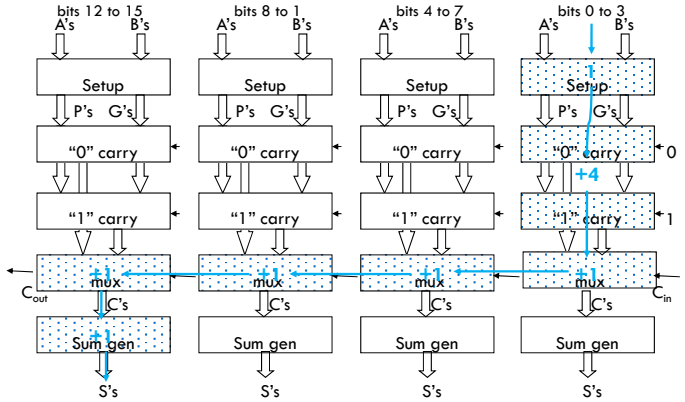


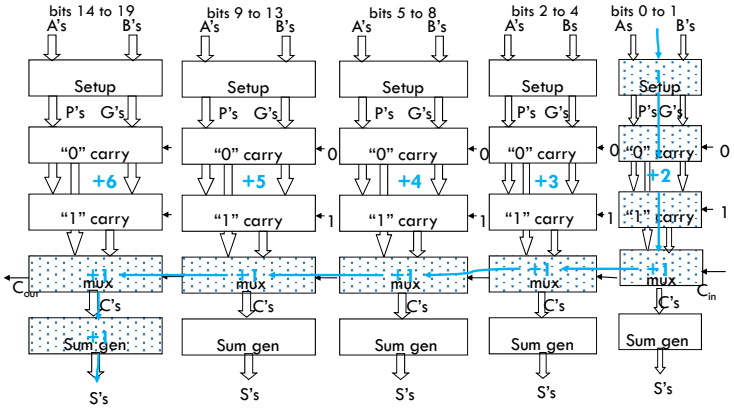
Figure 11.24 from Weste&Harris

Carry select adder: critical path



$T_{add} = t_{setup} + B\ t_{carry} + N/B\ t_{mux} + t_{sum}$

Square root carry select adder



$T_{add} = t_{setup} + 2\ t_{carry} + \sqrt{N}\ t_{mux} + t_{sum}$

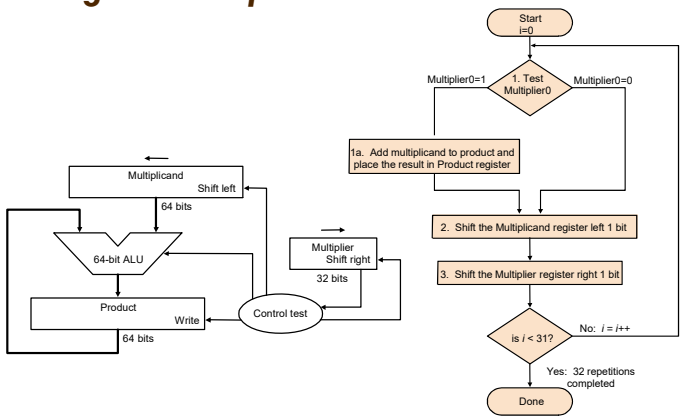
Integer multiplication revisited

- Right shift and add
  - Partial product rows accumulated from top to bottom on an *N*-bit adder
  - Time for *N* bits $T_{serial\_mult} = O(N\ T_{adder}) = O(N^2)$ for a RCA
- Making it faster
  - Use a faster adder
  - Use higher radix (e.g., base 4) multiplication
    - Use multiplier recoding to simplify multiple formation
  - Use carry-save-adders and avoid carry propagate at each cycle
  - Use multiple adders (array multiplier) with carry save adder cells.
    - Can be easily and efficiently pipelined
    - Very simple and efficient layout in VLSI

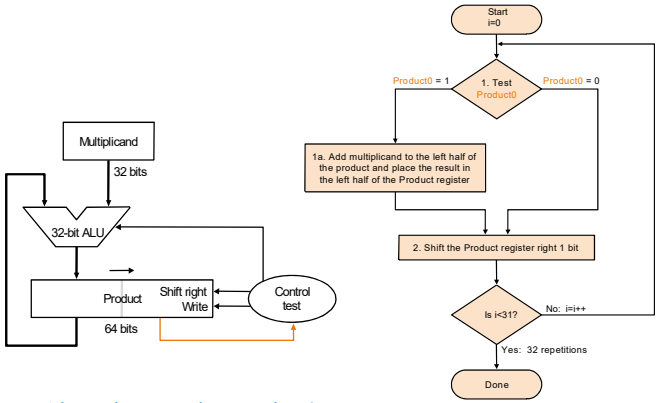
The binary multiplication

```
                1 0 1 0 1 0     Multiplicand
x                 1 0 1 1       Multiplier
-----------------------------
                1 0 1 0 1 0
              1 0 1 0 1 0
          0 0 0 0 0 0           Partial products
+       1 0 1 0 1 0             (can be formed in parallel)
-----------------------------
      1 1 1 0 0 1 1 1 0         Double Precision Result
```

Unsigned multiplication: version I



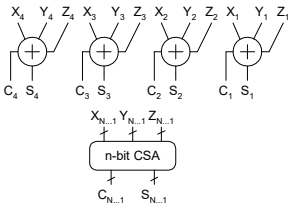
Unsigned multiplication, version II



What is the main advantage here?

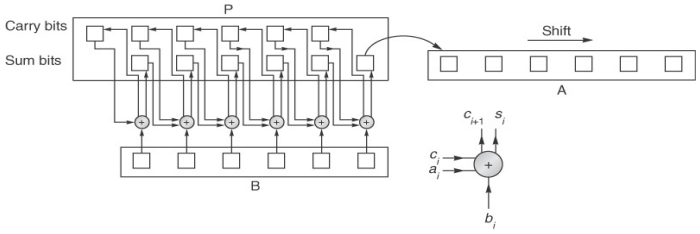
Carry Save Addition

- A full adder in the ith position sums 3 inputs and produces 2 outputs
  - Carry output belongs to $(i+1)^{th}$ bit, sum belongs to i^{th} bit
- N full adders in parallel are called *carry save adder (CSA)*
  - Produce N sums and N carry outs



BROWN

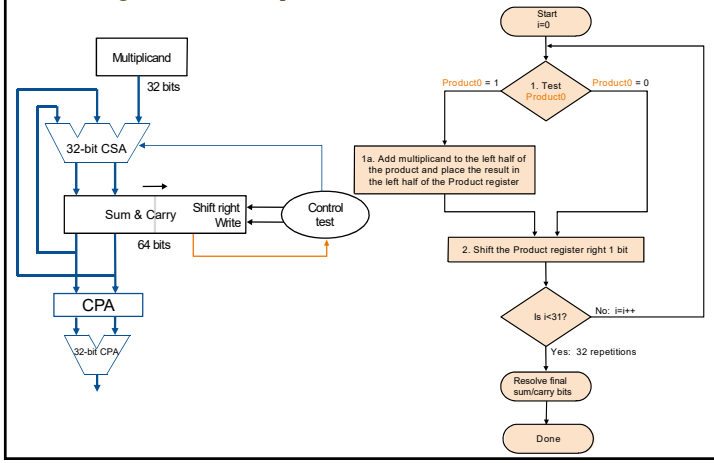
Carry-save multiplier



- A single carry-save adder is a collection of *n* independent adders
- Each addition results in a pair of bit vectors, C, S, stored separately in P
- The sum, carry bits of P are fed into the CSA in the following stage
- Requires a separate carry-propagate add at the end to combine the last carry, sum parts
- Still takes *n* cycles to compute, but each stage is faster.
  - Avoid waiting for carry to ripple through at each stage (save for later)

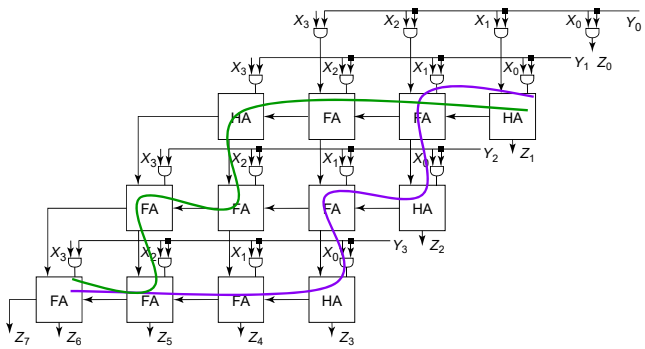
BROWN

Unsigned multiplication, version III



BROWN

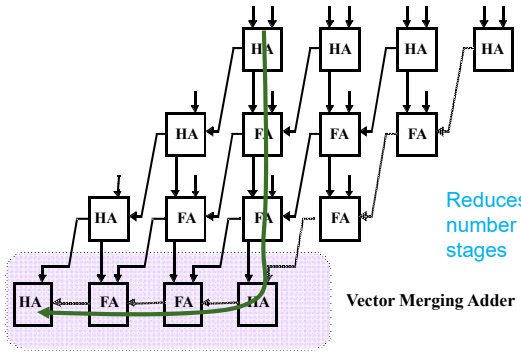
The Array Multiplier



Finding the critical path is not straightforward !

BROWN

Carry-save multiplier



Reduces the number of clock stages

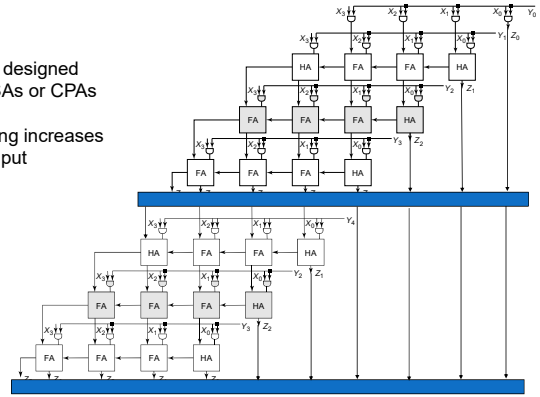
$t_{mult} = (N-1)t_{carry} + t_{and} + t_{merge}$

BROWN

Pipelined array multiplier

Can be designed with CSAs or CPAs

Pipelining increases throughput



Power Dissipation Revisited

It's not just about dynamic power dissipation

BROWN

BROWN

CMOS energy & power equations

$$E = C_L V_{DD}^2 P_{0\to1} + t_{sc} V_{DD} I_{peak} P_{0\to1} + V_{DD} I_{leakage}$$

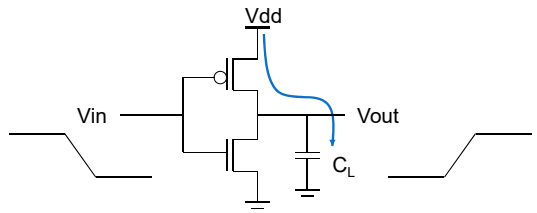
$$f_{0\to1} = P_{0\to1} * f_{clock}$$

$$P = C_L V_{DD}^2 f_{0\to1} + t_{sc} V_{DD} I_{peak} f_{0\to1} + V_{DD} I_{leakage}$$

Dynamic power | Short-circuit power | Leakage power

BROWN

Dynamic power dissipation



$$\text{Energy/transition} = C_L * V_{DD}^2 * P_{0\to1}$$

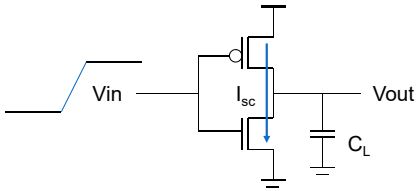
$$P_{dyn} = \text{Energy/transition} * f = C_L * V_{DD}^2 * P_{0\to1} * f$$

$f_{0\to1}$

$$P_{dyn} = C_{EFF} * V_{DD}^2 * f \quad \text{where } C_{EFF} = P_{0\to1}\, C_L$$

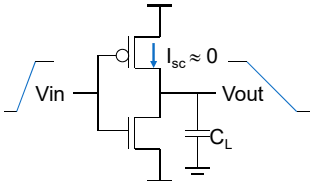
Data dependent → a function of switching activity!

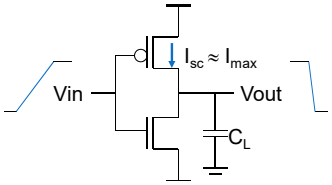
Short circuit power dissipation



Finite slope of the input signal causes a direct current path between V_{DD} and GND for a short period of time during switching when both the NMOS and PMOS transistors are conducting.

Impact of C_L on P_{sc}

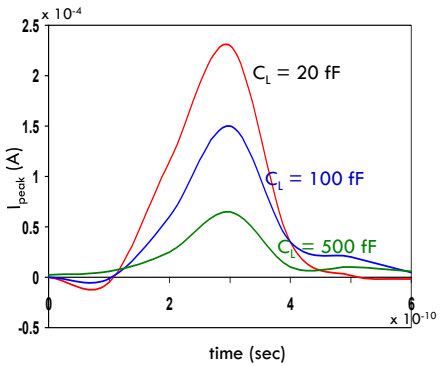




Large capacitive load
BUT BOTH DEVICES ARE ON FOR A SHORT TIME
Output fall time significantly larger than input rise time.

Small capacitive load
BOTH DEVICES ARE ON FOR A LONG TIME
Output fall time substantially smaller than the input rise time.

I_{peak} as a function of C_L

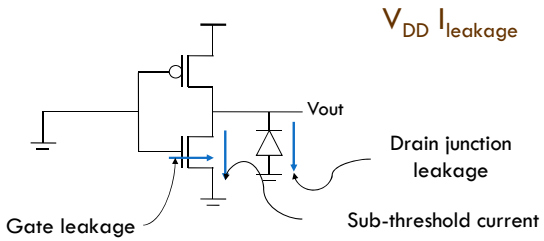


500 psec input slope

When load capacitance is small, I_{peak} is large.

Short circuit dissipation is minimized by matching the rise/fall times of the input and output signals - slope engineering.

Leakage (static) power dissipation



Sub-threshold current is the dominant factor.

All increase exponentially with temperature!

BROWN

Reducing static power

$$I_D \approx I_S e^{\frac{V_{GS}-V_{th}}{nkT/q}}\left(1-e^{-\left(\frac{V_{DS}}{kT/q}\right)}\right)$$

- V_{th} has an exponential effect on leakage current
- Increasing V_{th} will lower static power dissipation
  - Control indirectly (taking advantage of circuit topology and input values)
  - Control directly (using low V_{th} devices)

BROWN

Stack effect

- Leakage is a function of the circuit topology and the value of the inputs
- $I_{leakage}$ is mainly a function of V_{GS} and V_{th} (which is a function of V_{SB}).

$$V_{th} = V_{T0} + \gamma\left(\sqrt{|-2\varphi_F + V_{SB}|} - \sqrt{|-2\varphi_F|}\right)$$

where V_{T0} is the threshold voltage at $V_{SB} = 0$; V_{SB} is the source- bulk (substrate) voltage; γ is the body-effect coefficient; ϕ_F is the Fermi potential (negative for NMOS)

- For an NMOS device, as V_{SB} increases (i.e. becomes positive), V_{th} *increases.*

BROWN

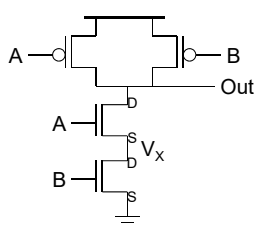
Stack Effect

$$V_{th} = V_{T0} + \gamma\left(\sqrt{|-2\varphi_F + V_{SB}|} - \sqrt{|-2\varphi_F|}\right)$$

- Consider a 2-input NAND gate
- Leakage is lowest when both inputs A=B=0V.
  - Intermediate node V_x settles to:

$$I_D \approx I_S e^{\frac{V_{GS}-V_{th}}{nkT/q}}\left(1-e^{-\left(\frac{V_{DS}}{kT/q}\right)}\right)$$

$V_x \approx V_{th} \ln(1+n)$



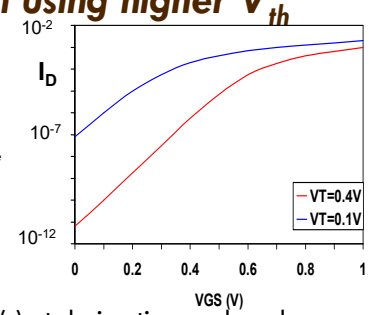
A	B	V_X	$I_{SUB}=f(I_s, V_{GS}, V_{BS})$
0	0	$V_{th}\ ln(1+n)$	$V_{GS}=V_{BS}= -V_X$
0	1	0	$V_{GS}=V_{BS}=0$
1	0	$V_{DD}-V_T$	$V_{GS}=V_{BS}=0$
1	1	0	$V_{SG}=V_{SB}=0$

Leakage reduction due to stacked transistors is called the stack effect

BROWN

Leakage reduction using higher V_{th}

- Reducing V_{th} increases $I_{leakage}$ (exponentially)
- But, reducing V_{th} decreases gate delay

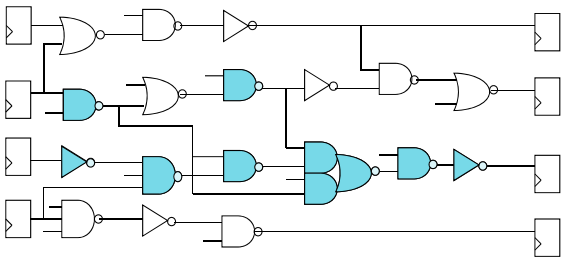


➡ Determine the critical path(s) at design time and use low V_{th} devices for transistors from gates on critical paths. Use a high V_{th} on other logic for leakage control.

- A careful assignment of V_T can reduce the leakage significantly

Dual-Thresholds inside a logic block

- Minimum energy consumption is achieved if all logic paths are critical (have the same delay)
- Use lower threshold on timing-critical paths
- *Is there an advantage of dual-V_t over dual V_{DD}?*



Dynamic power is data dependent

- Switching activity, $P_{0\to1}$, has two components
  - A static component : function of logic topology
  - A dynamic component : function of the timing behavior (glitching)

2-input NOR Gate

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

Static transition probability

$P_{0\to1} = P_{out=0} \times P_{out=1}$

$= P_0 \times (1-P_0)$

With input signal probabilities

$P_{A=1} = 1/2$

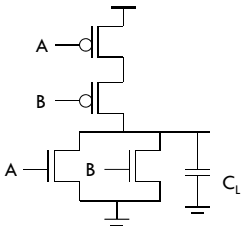
$P_{B=1} = 1/2$

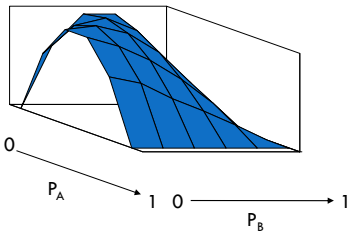
NOR static transition probability

$= 3/4 \times 1/4 = 3/16$

NOR Gate Transition Probabilities

- Switching activity is a strong function of the input signal statistics
  - P_A and P_B are the probabilities that inputs A and B are one

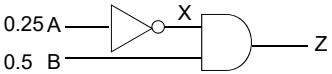




$P_{0\to1} = P_0 \times P_1 = (1-(1-P_A)(1-P_B))\,(1-P_A)(1-P_B)$

Transition Probabilities for Some Basic Gates

	$P_{0\to1} = P_{out=0} \times P_{out=1}$
NOR	$(1 - (1 - P_A)(1 - P_B)) \times (1 - P_A)(1 - P_B)$
OR	$(1 - P_A)(1 - P_B) \times (1 - (1 - P_A)(1 - P_B))$
NAND	$P_AP_B \times (1 - P_AP_B)$
AND	$(1 - P_AP_B) \times P_AP_B$
XOR	$(1 - (P_A + P_B - 2P_AP_B)) \times (P_A + P_B - 2P_AP_B)$



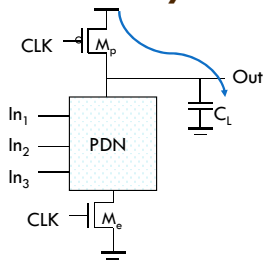
For X: $P_{0\to1} = P_0 \times P_1 = (1-P_A)\,P_A$

$= 0.75 \times 0.25 = 0.1875$

For Z: $P_{0\to1} = P_0 \times P_1 = (1-P_XP_B)\,P_XP_B$

$= (1 - (0.75 \times 0.5)) \times (0.75 \times 0.5) = 0.1523$

Power dissipation of dynamic gate



Power only dissipated when previous Out = 0

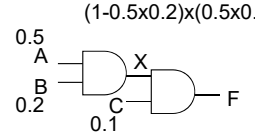
$$E = C_L V_{DD}^2 P_{0\to1}$$

Probability the output transitions from 0 to 1

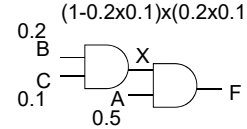
Switching activity can be higher in dynamic gates!

Input Ordering

(1-0.5x0.2)x(0.5x0.2)=0.09



(1-0.2x0.1)x(0.2x0.1)=0.0196

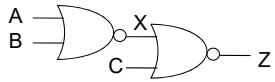


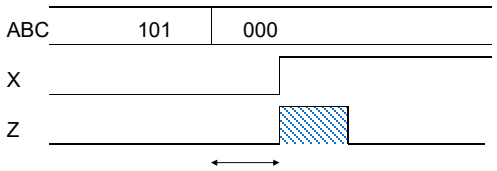
Beneficial to postpone the introduction of signals with a high transition rate (signals with signal probability close to 0.5)

What about activity factor at node F for each circuit?

Glitching in Static CMOS Networks

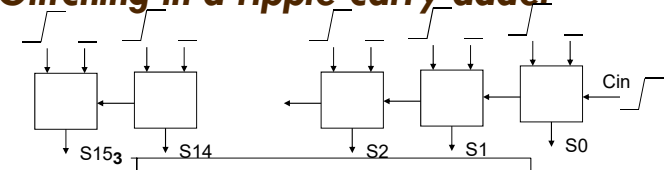
- Gates have a nonzero propagation delay resulting in spurious transitions or glitches (dynamic hazards)
  - glitch: node exhibits multiple transitions in a single cycle before settling to the correct logic value

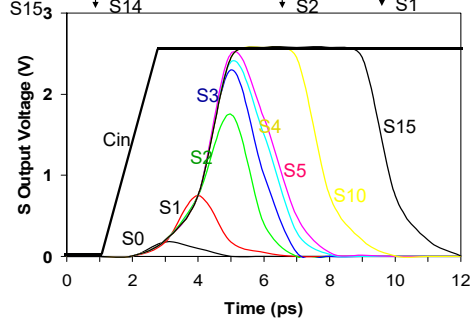




Unit Delay

Glitching in a ripple carry adder

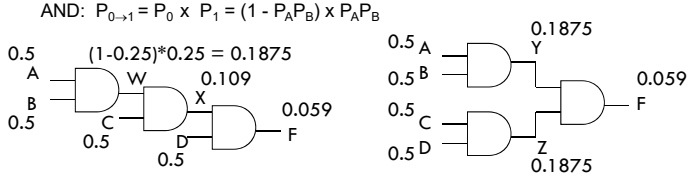




Logic Restructuring

- Change the topology of a logic network to reduce transitions

AND: $P_{0\to1} = P_0 \times P_1 = (1 - P_A P_B) \times P_A P_B$

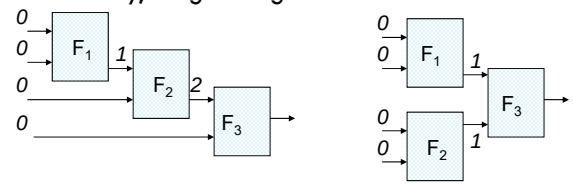


- Chain implementation has a lower overall switching activity than the tree implementation for random inputs

Ignores glitching effects

Balanced delay paths to reduce glitching

- Glitching is due to a mismatch in the path lengths in the logic network; if all input signals of a gate change simultaneously, no glitching occurs



- So equalize the lengths of timing paths through logic