



BROWN
School of Engineering

DIGITAL ELECTRONICS SYSTEM DESIGN

FALL 2019

PROF. IRIS BAHAR

NOVEMBER 18, 2019

LECTURE 20: FAST ADDITION,

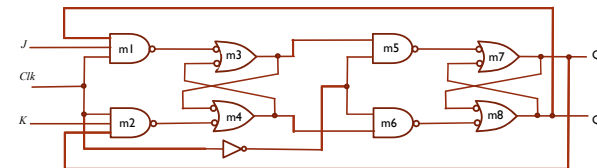
GROUP 2 LAB EXTENSION

- I have decided to extend the deadline for group 2 labs to Wednesday, Nov. 20 at 6:30pm
- Please note that you should not expect similar extensions for groups 3 and 4. Please plan your time accordingly
- For lab 7, please note that now is a good time to make use of the logic analyzer for your debugging, if you haven't done so already

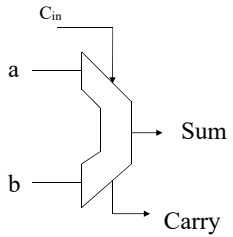
MIDTERM EXAM

- The exam has been graded. Here are some stats:
 - Average: 68.5
 - Min: 35
 - Max: 92
 - Median: 69
- Here is a *rough* grade distribution:
 - A: ≥ 76
 - A/B: $70 \leq B \leq 75$
 - B: $60 \leq B \leq 69$
 - B/C: $55 \leq B \leq 59$
 - C: $40 \leq B \leq 54$
 - F: < 40

PROBLEM 3B



FULL ADDER

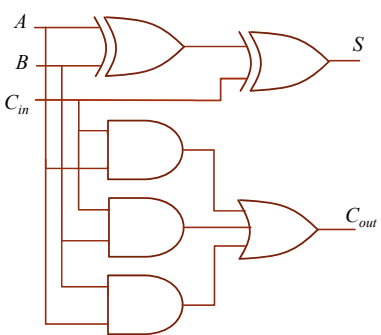


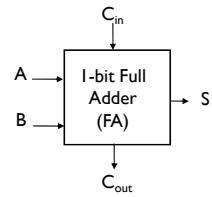
Truth Table

Id	a	b	c_{in}	carry	sum
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	1

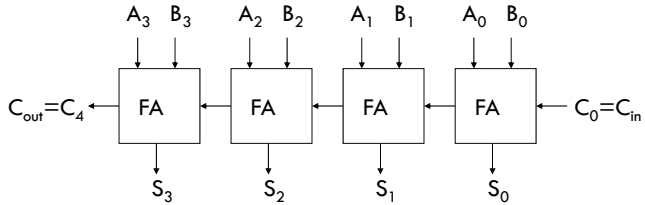
■ How do you express sum and carry as Boolean functions?

THE FULL ADDER



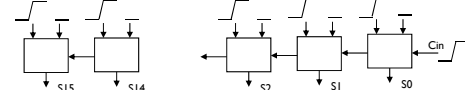


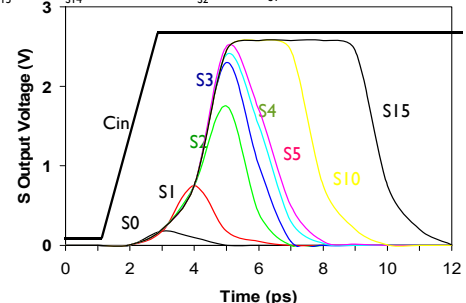
THE RIPPLE CARRY ADDER



■ The carry out of one stage ripples to the carry in of the next

GLITCHING IN A RIPPLE CARRY ADDER





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 $G_i = A_i \& B_i = A_i B_i$
 - propagated $P_i = A_i \oplus B_i$
 - annihilated (killed) $K_i = !A_i \& !B_i$
- Giving a carry recurrence of

$$C_{i+1} = G_i + P_i C_i$$

$$C_1 = G_0 + P_0 C_0$$

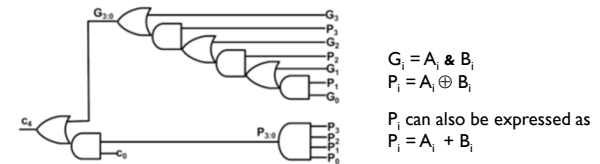
$$C_2 = G_1 + P_1 C_1 = G_1 + P_1 G_0 + P_1 P_0 C_0$$

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

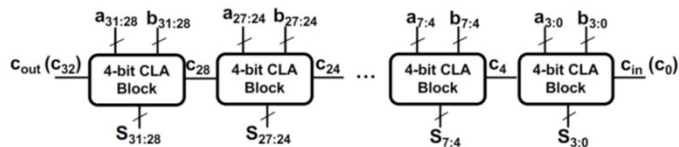
$$C_4 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$$

CARRY OUT LOGIC FOR 4-BIT ADDER

- P_i s and G_i s are computed at time 0
- $C_4 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$
 $= G_3 + P_3(G_2 + P_2(G_1 + P_1 G_0)) + (P_3 P_2 P_1 P_0) C_0$

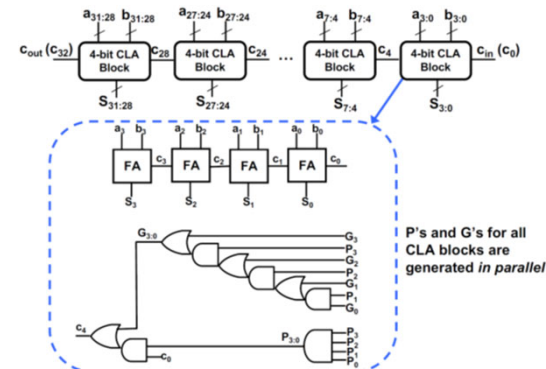


32-BIT CARRY LOOKAHEAD ADDER WITH 4-BIT RIPPLE CARRY ADDERS

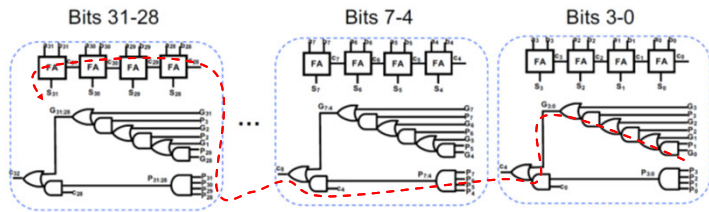


- Carry out logic gets more complicated beyond 4 bits
- CLAs are often implemented as 4-bit modules and instantiated in a hierarchical way to realize wider adders

32-BIT CLA WITH 4-BIT RCAS

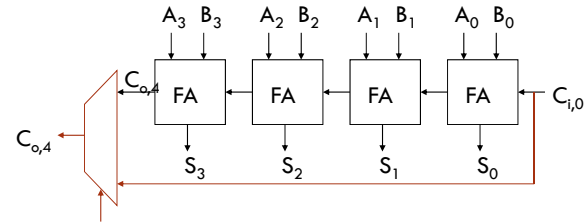


WORST CASE DELAY



- What is the longest delay to compute the full add?

CARRY-SKIP (CARRY-BYPASS) ADDER

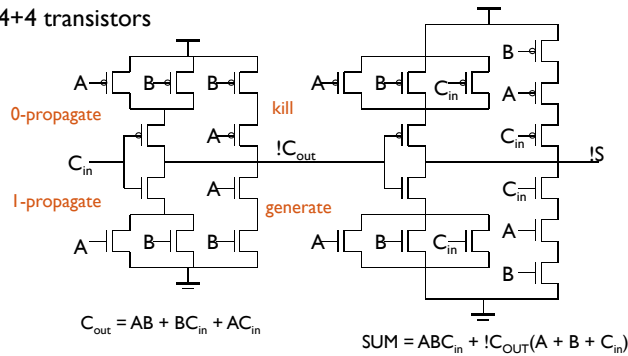


$$BP = P_0 P_1 P_2 P_3 \text{ "Block Propagate"}$$

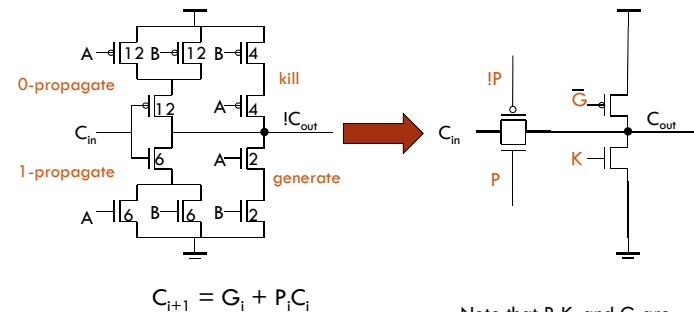
If $(P_0 \& P_1 \& P_2 \& P_3 = 1)$ then $C_{o,4} = C_{i,0}$ otherwise the block itself kills or generates the carry $C_{o,4}$ (and doesn't need $C_{i,0}$ to compute $C_{o,4}$)

MIRROR ADDER

24+4 transistors



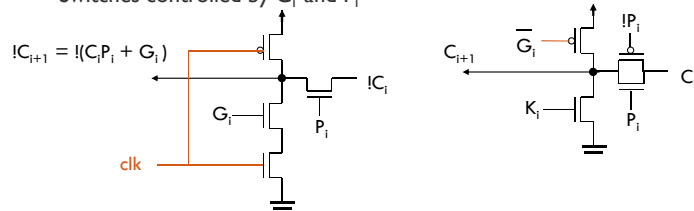
MIRROR ADDER



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

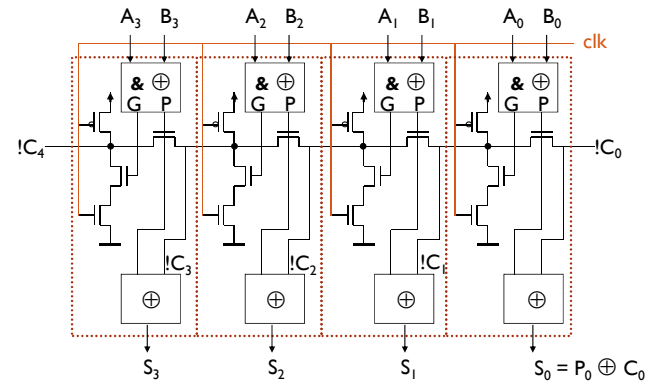
MANCHESTER CARRY CHAIN

- Switches controlled by G_i and P_i

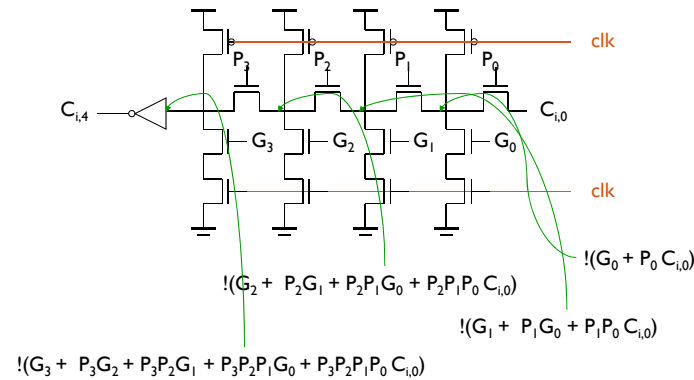


- Total delay of
 - time to form the switch control signals G_i and P_i
 - setup time for the switches
 - signal propagation delay through N switches in the worst case

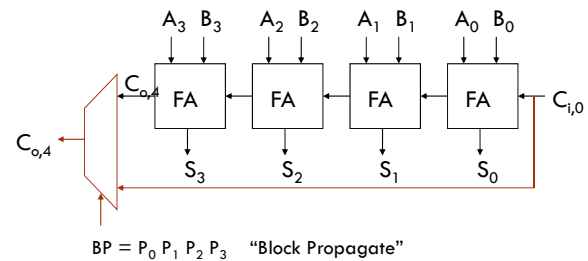
4-BIT SLICED MCC ADDER



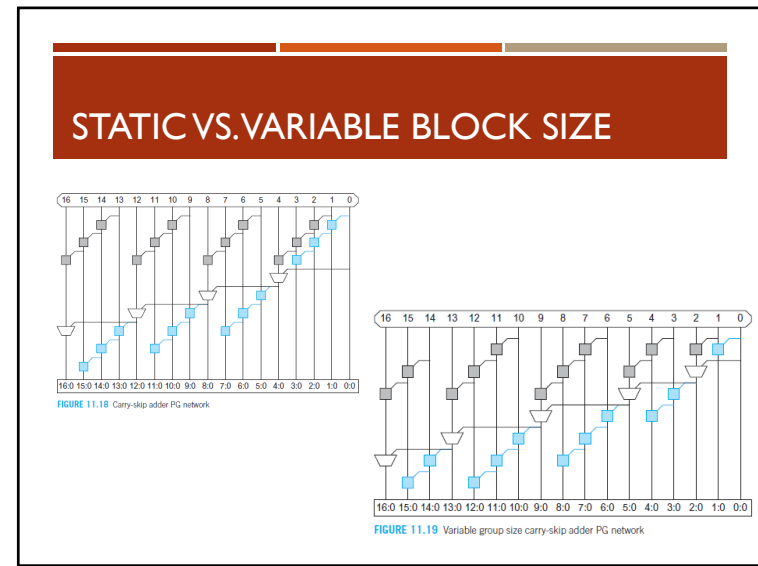
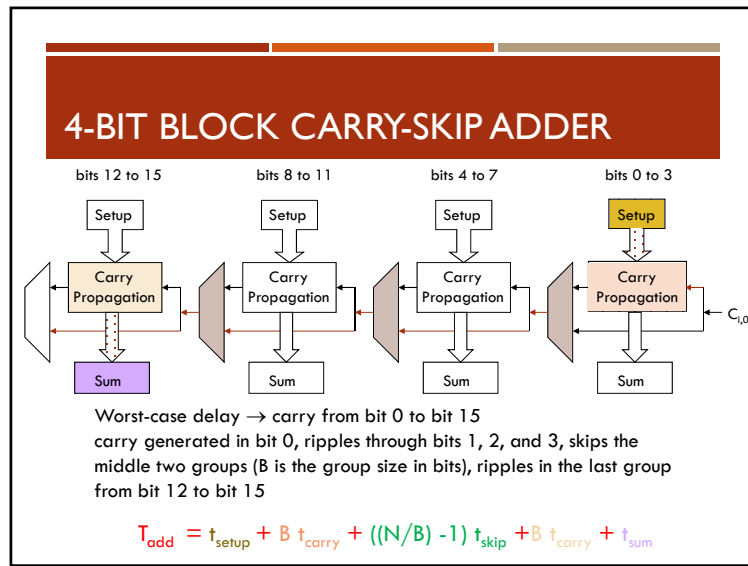
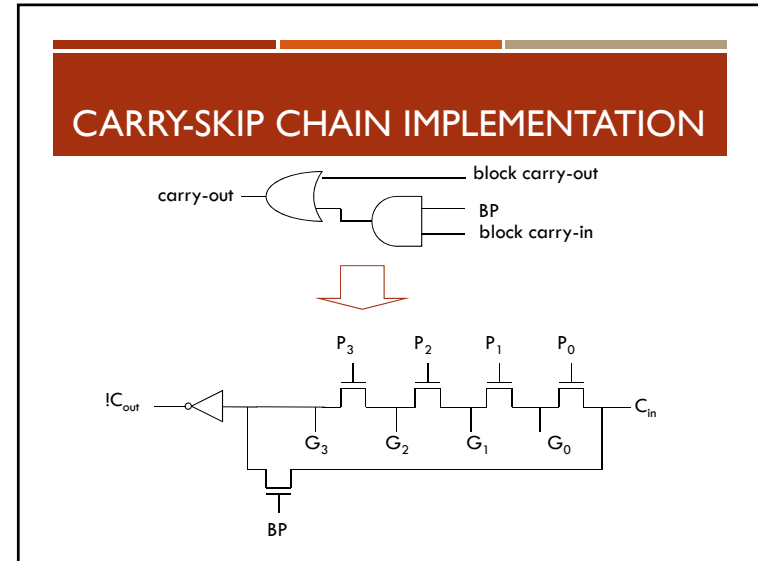
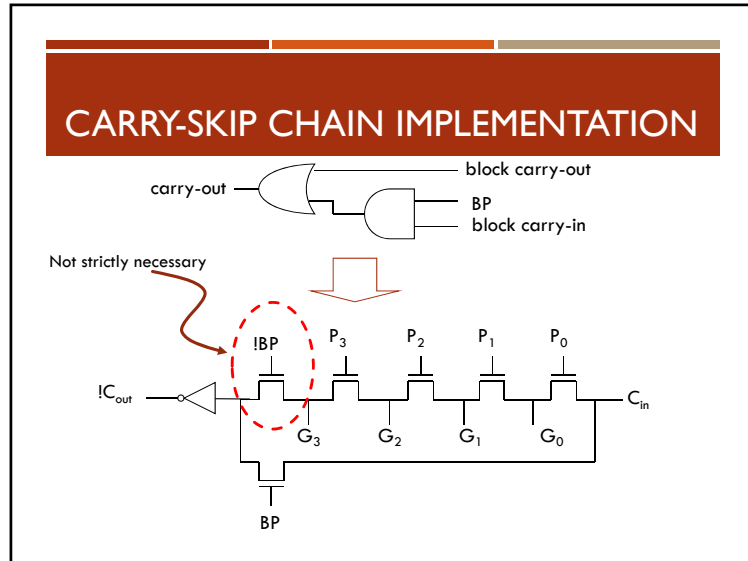
DOMINO MANCHESTER CARRY CHAIN



CARRY-SKIP (CARRY-BYPASS) ADDER

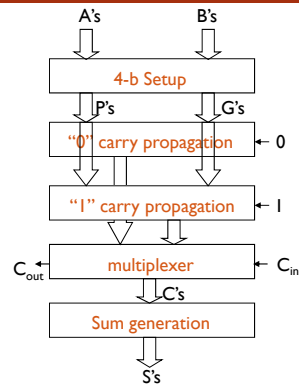


If $(P_0 \& P_1 \& P_2 \& P_3 = 1)$ then $C_{o,4} = C_{i,0}$ otherwise the block itself kills or generates the carry internally (and doesn't need $C_{i,0}$ to compute $C_{o,4}$)



CARRY 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 select "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.

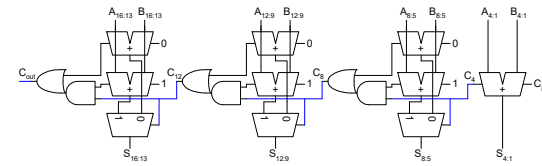
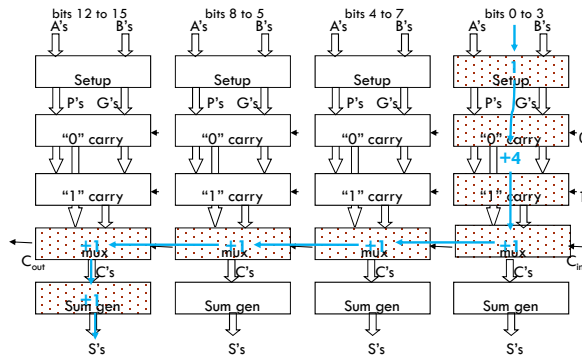


Figure 11.24 from Weste&Harris

CARRY SELECT ADDER: CRITICAL PATH



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