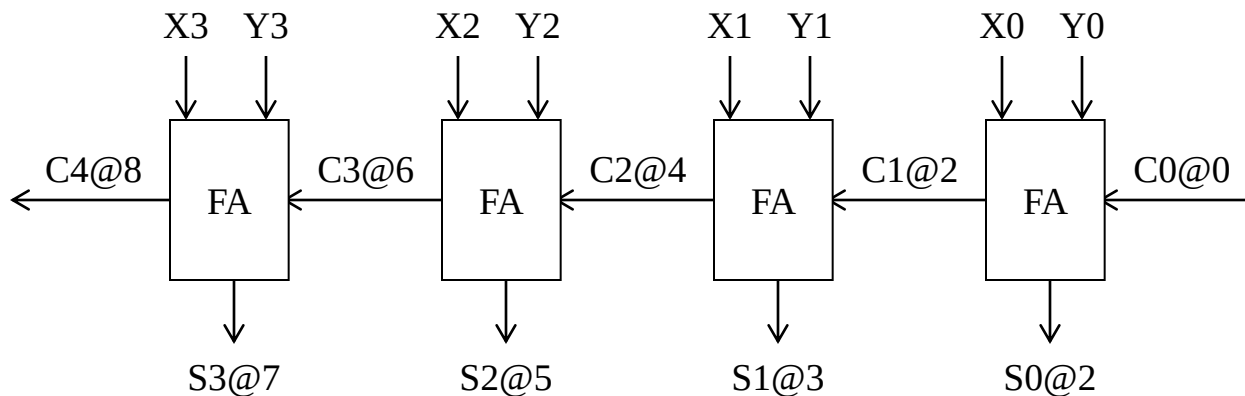
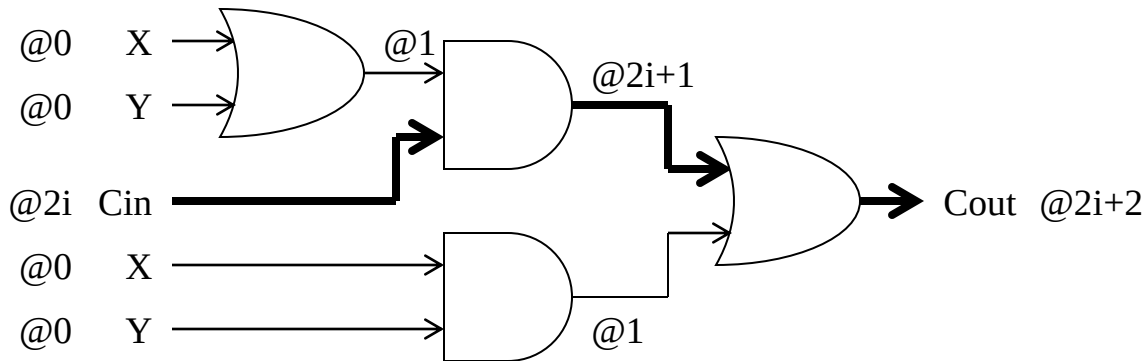


Fast Adders

Ripple-Carry Adder Critical Delay

- Delay grows to $2N + 2$, where N = number of bits. Problem when N is large, say $N = 16$.



Carry Lookahead Adder

- Normally
$$S_i = X_i \oplus Y_i \oplus C_i$$
$$C_{i+1} = X_i Y_i + C_i (X_i \oplus Y_i)$$
- We can define **Propagate** (P) and **Generate** (G) functions as follows:
$$P_i = X_i \oplus Y_i$$
$$G_i = X_i Y_i$$
$$S_i = P_i \oplus C_i$$
$$C_{i+1} = G_i + P_i C_i$$
- Then
$$C_1 = G_0 + P_0 C_0$$
$$C_2 = G_1 + P_1 C_1$$
$$C_3 = G_2 + P_2 C_2$$
$$C_4 = G_3 + P_3 C_3$$

Carry Lookahead Adder

- We can “flatten” the following equations

$$C_1 = G_0 + P_0 C_0$$

$$C_2 = G_1 + P_1 C_1$$

$$C_3 = G_2 + P_2 C_2$$

$$C_4 = G_3 + P_3 C_3$$

- To obtain

$$C_1 = G_0 + P_0 C_0$$

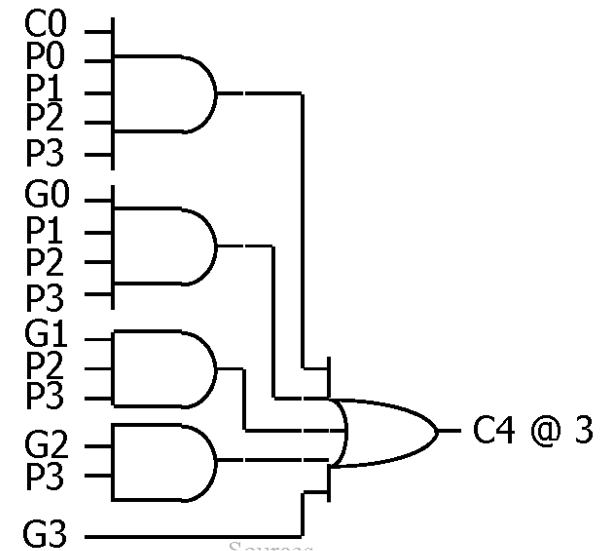
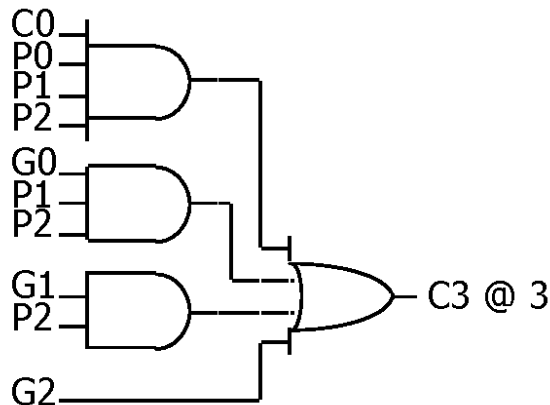
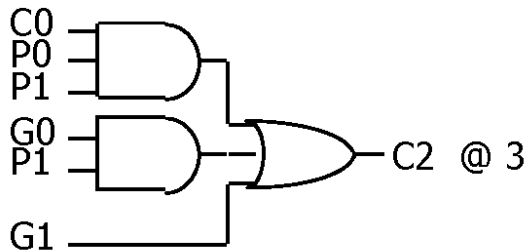
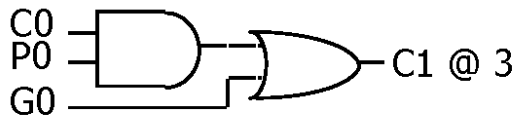
$$C_2 = G_1 + G_0 P_1 + C_0 P_0 P_1$$

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

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

Carry Lookahead Adder

- Once “flattened”, all C_i 's can be computed with @3 delay.
- Since $S_i = P_i \oplus C_i$, all S_i 's can be computed with @4 delay.
- Problem: the AND and OR gates get increasingly large w.r.t. “i”. i.e., difficult to implement larger than 5-input AND or 5-input OR.



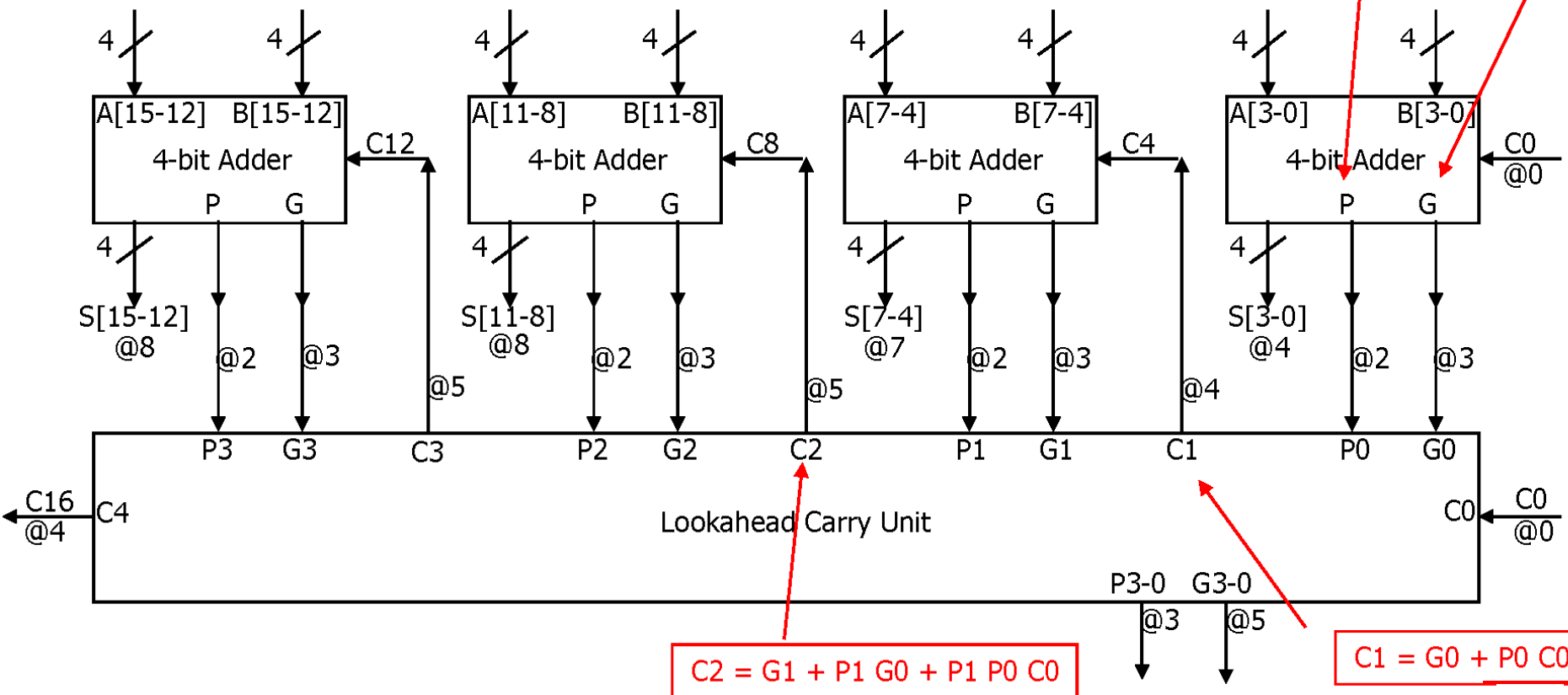
Carry Lookahead Adder

- Cascaded carry lookahead logic

- Four 4-bit adders with internal carry lookahead
- Second-level carry lookahead unit extends lookahead to 16 bits
- S15@8 for carry-lookahead vs. $2(15) + 2 = @32$ for C16 of ripple-carry adder

$$G = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0$$

$$P = P_3 P_2 P_1 P_0$$



Carry Select Adder

- Redundant hardware to make carry calculation go faster
 - Compute two high-order sums in parallel while waiting for carry-in
 - One assuming carry-in = 0, another assuming carry-in = 1
 - Select correct result once carry-in is finally computed

