CSE4210 Architecture and Hardware for DSP

Lecture 2

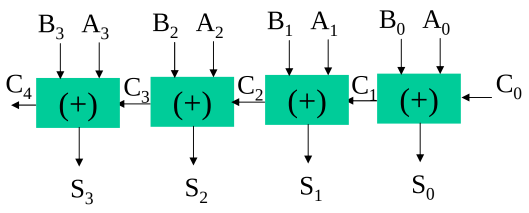
Fast Addition

Fast Addition

- Carry-Lookahead Adders
- Carry Select Adders
- Conditional Sum Adders
- Carry Save Adders

York University CSE

Ripple Carry Adder



York University CSE

Carry Lookahead Adders

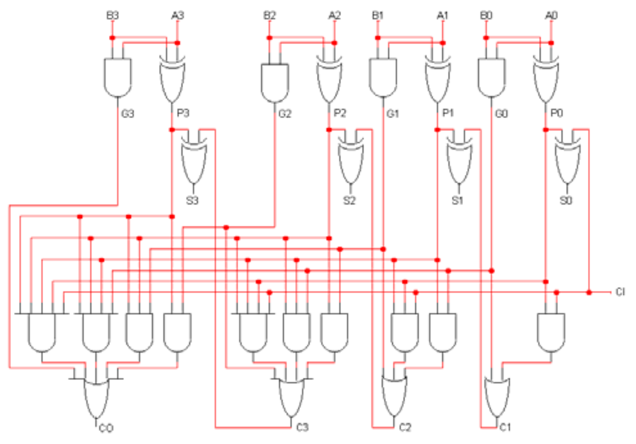
$$G_i = A_i \bullet B_i$$
$$P_i = A_i \oplus B_i$$
$$S_i = P_i \oplus C_i$$
$$C_{i+1} = G_i + P_i \bullet C_i$$
$$C_1 = G_0 + C_0 P_0$$
$$C_2 = G_1 + C_1 P_1 = G_1 + G_0 P_1 + C_0 P_0 P_1$$
$$C_3 = G_2 + C_2 P_2 = G_2 + G_1 P_1 + G_0 P_1 P_2 + C_0 P_0 P_1 P_2$$

$$C_n = G_{n-1} + G_{n-2} P_{n-1} + \cdots + G_0 P_1 P_2 \cdots P_{n-1} + C_0 P_0 P_1 \cdots P_{n-1}$$

Carry Lookahead Adders

- Time = Carry generate/propagate + carry lookahead (to generate C's)+ sum (EXOR) = 3+2+2=7 gate delays.
- The problem is we need n-input AND, for 32 or 64-bit numbers that is not practical.

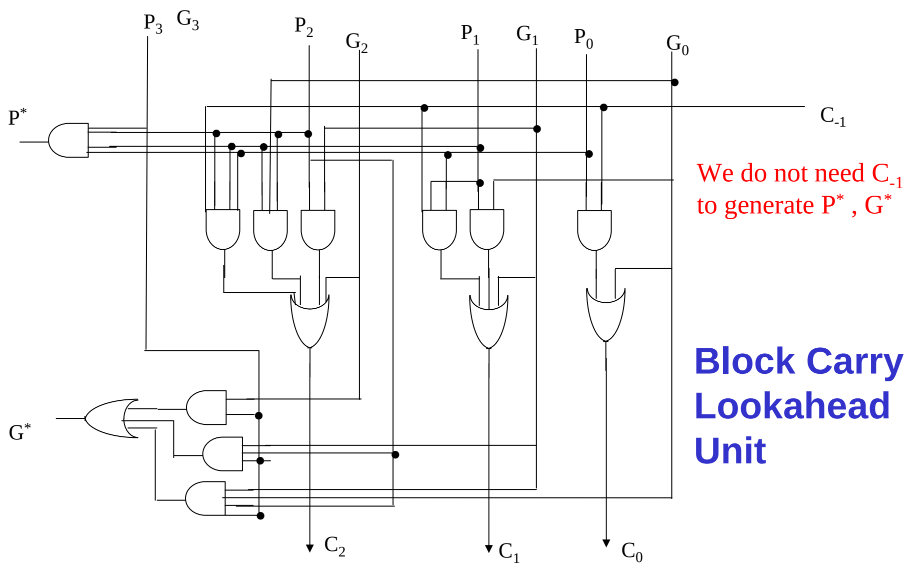
Carry Lookahead Adders



Carry Lookahead Adders

- In order to limit the fan-in in the carry generation circuitry, we use block carry lookahead adders.
- Consider for examples the number is divided into blocks each of 4 bits.
- That means the fan-in will be limited to 4 (not 5)
- Each block has a *block carry generate* G* and *block carry propagate* P*

Carry Lookahead Adders

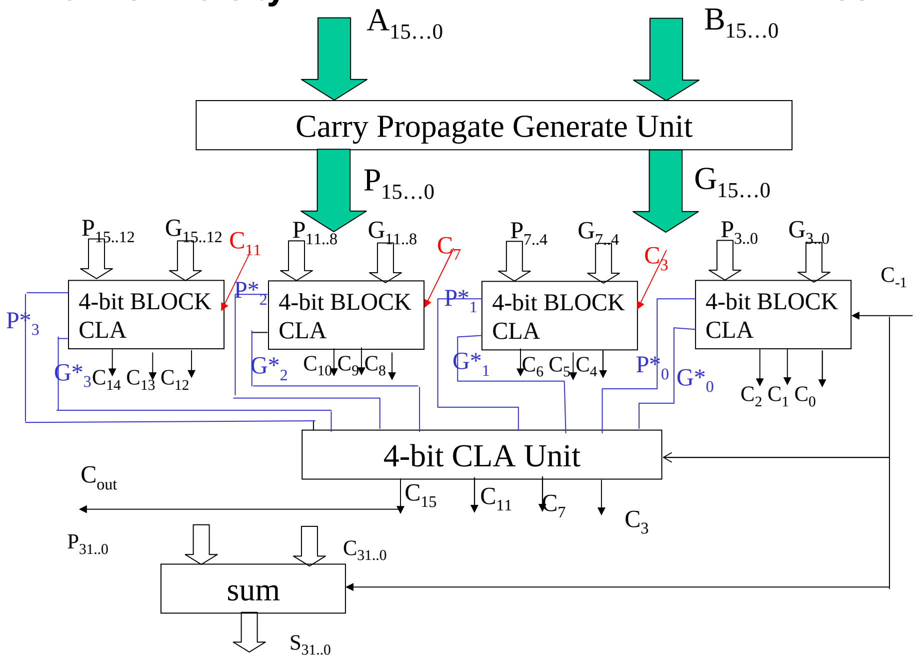


We do not need C_{-1} to generate P^* , G^*

Block Carry Lookahead Unit

Carry Lookahead Adders

- Consider 16-bit numbers divided into 4 blocks of 4 bit each



Carry Select Adders

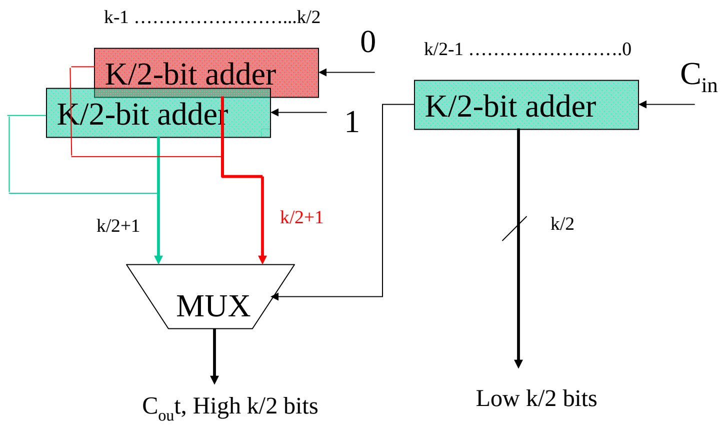
- The idea is instead of waiting for the carry from the right, we compute two sums.
  - One assuming the carry is 0
  - The other assuming the carry is 1
- Once we know the carry, then we can select the right sum using a mux

Carry Select Adders

- Can combine 3 k/2-bit adders to produce one k-bit adder.
- One k/2-bit adder is used to compute the lower half of the k-bit sum.
- The other 2 k/2-bit adders are used to compute the higher sum assuming 1 or 0 carry.
- Once we know the carry, choose the right higher order k/2 bits

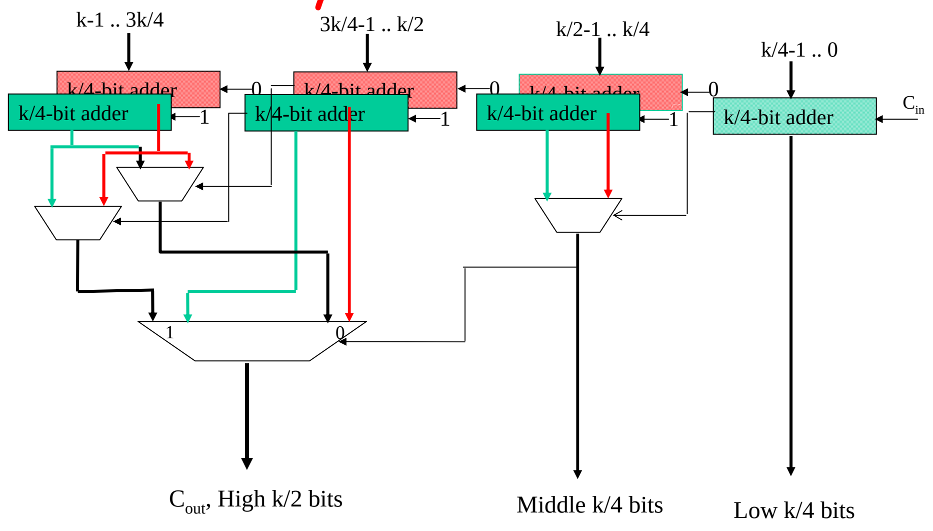
York University **CSE**

Carry Select Adders



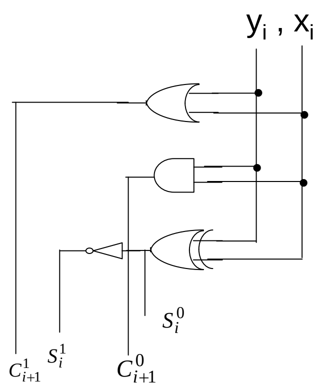
York University **CSE**

Carry Select Adders



Carry Select Adders

- What is the time and cost of CSA?



Conditional Sum Adders

- Conditional sum adders use the same idea as carry select adders by continuing the decomposition of the numbers until we reach a one bit.
- See the blackboard for an example (too big to fit in one page here).

Carry Save Adders

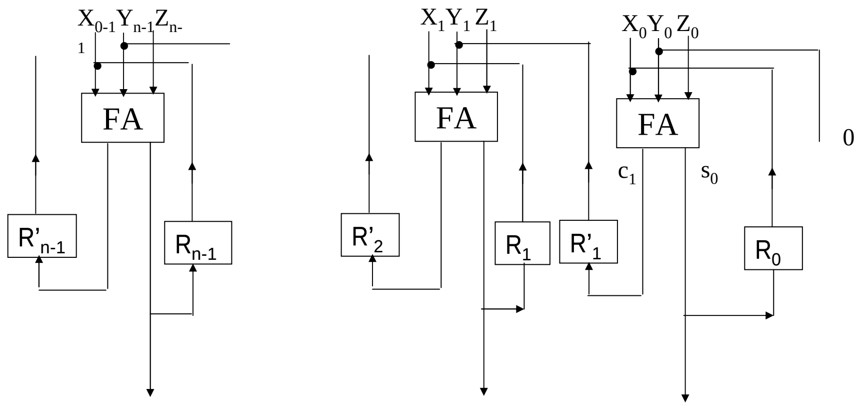
- So far, we mentioned adders that add 2 numbers.
- In multiplication (and accumulation) we may want to add more than 2 numbers.
- Instead of waiting for the carry propagation of the first addition, to be completed, before starting the second addition, we overlap the carry of the first addition with the computation of the second addition. ….

Carry Save Adders

- After the last addition, we have to wait for the carry propagation.
- For large values of *n* the carry propagation after the last addition may be too long.
- We can use any previously studied methods in order to speed up the operation.

York University CSE

Carry Save Adders



York University CSE

Carry Save Adders

1. Input f_i^1, f_i^2, f_i^3 to the full adder and add, store s_i in R_i and c_{i+1} in R'_I
2. Feed s_i back to the full adder in the current bit position, and c_{i+1} to the adder in position i+1, and add f^4
3. Repeat 2 (enter the f^5 till f^k)until all the numbers are added
4. Repeat 3 until all the R' are 0

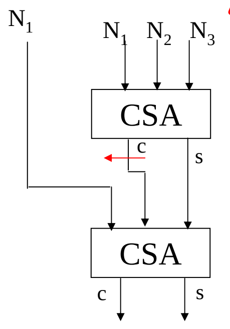
York University CSE

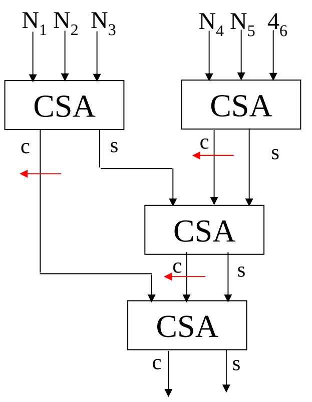
Multilevel Carry-Save Adders

- The previous design still inputs one additional number per cycle.
- CSA tree can be sued to add k numbers simultaneously.
- Two feedback inputs in each case are needed to accumulate the *partial sum* and the *one-bit left-shifted partial carry*

York University CSE

Multilevel Carry-Save Adders



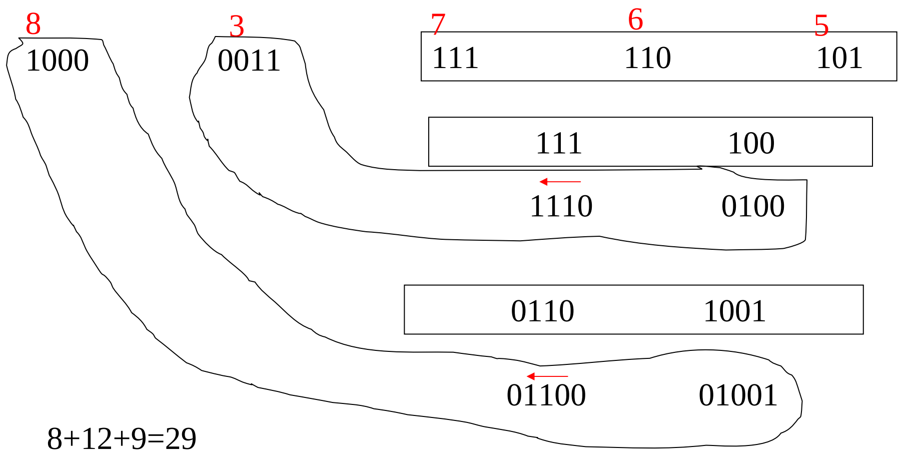


← On the carry output lines indicate shift left one bit with 0 entering from the right

Each CSA (except the last one) is n-full adders

York University CSE

Multilevel Carry-Save Adders



8+12+9=29