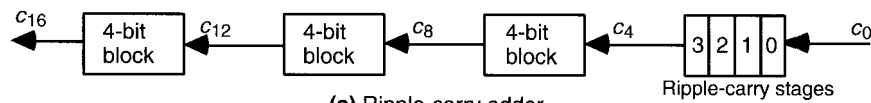


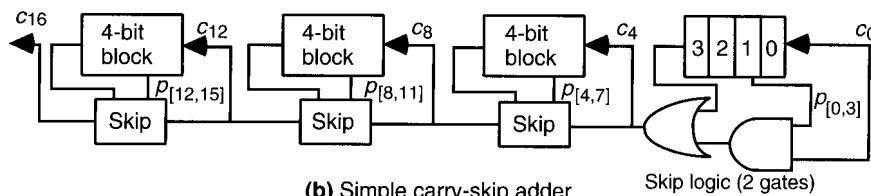
Lecture 7

Variations in Fast Adders

Simple Carry-Skip Adders



(a) Ripple-carry adder.



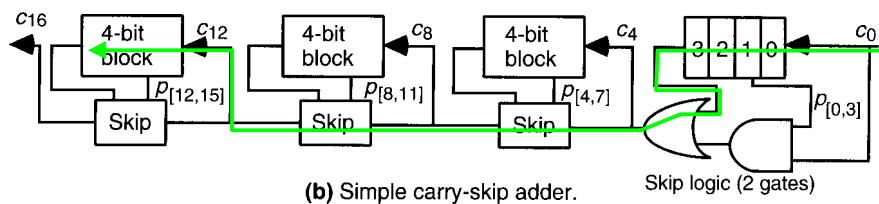
(b) Simple carry-skip adder.

Simplifying Assumptions

- One skip delay (c_{in} to c_{out}) is equal to one ripple delay (c_{in} to c_{out})
- Total k -bit ripple delay is $k \times$ single delay

These assumptions may not be true in real life (CMOS implementation for example).

Worst Case Delay



b = fixed block width (ex : 4)

k = number of bits (ex : 16)

$$T_{delay} = \underbrace{(b-1)}_{\text{block 0}} + \underbrace{(0.5)}_{\text{OR-gate}} + \underbrace{(k/b-2)}_{\text{skips}} + \underbrace{(b-1)}_{\text{block (k-1)}}$$

$$\approx 2b + k/b - 3.5 \text{ stages (ex : 12.5)}$$

What is the optimal block size?

1. set $\frac{dT}{db} = 0$

2. solve for $b = b_{opt}$

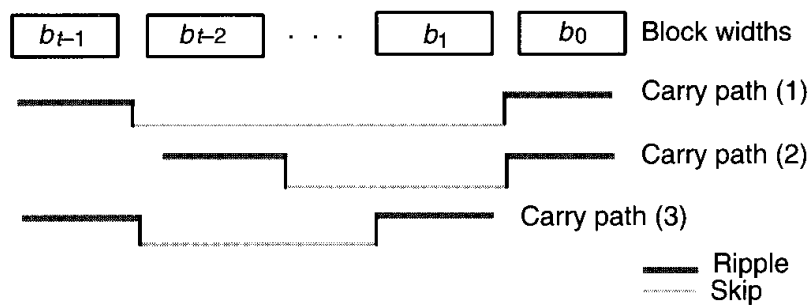
$$b_{opt} = \sqrt{\frac{k}{2}}$$

$$t = \text{number of blocks} = k/b$$

$$t_{opt} = \sqrt{2k}$$

$$T_{opt} = 2\sqrt{2k} - 3.5$$

Can we do better?

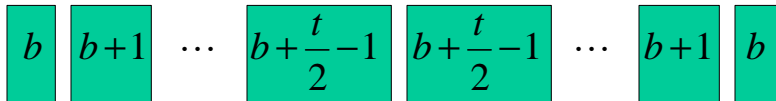


Path (1) is one delay longer than Path (2) →
block $t-2$ can be one bit wider than block $t-1$.

Path (1) is one delay longer than Path (3) →
block 1 can be one bit wider than block 0 .

Variable Block-Width Carry-Skip Adders

Optimal Block Widths :



$$b + (b+1) + \dots + (b + \frac{t}{2} - 1) + (b + \frac{t}{2} - 1) + \dots + (b+1) + b = k$$

$$\rightarrow b = (k/t) - (t/4) + 1/2$$

Optimal number of blocks

$$T_{delay} = \underbrace{2(b-1)}_{\text{first + last stage}} + \underbrace{(0.5)}_{\text{OR gate}} + \underbrace{(t-2)}_{\text{Skip stages}} = \frac{2k}{t} + \frac{t}{2} - 2.5$$

1. set $\frac{dT}{dt} = 0$

2. solve for $t = t_{opt}$

$$t_{opt} = 2\sqrt{k}$$

$$b_{opt} = \lceil 1/2 \rceil = 1 \quad (\text{stage } 0, t-1; \text{ goes up to } t_{opt}/2 = \sqrt{k})$$

$$T_{opt} = 2\sqrt{k} - 2.5$$

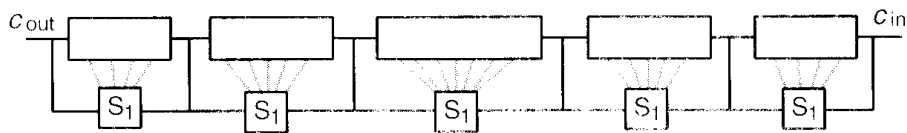
Comparison

	Fixed-width Carry-Skip	Variable-width Carry-Skip
t_{opt}	$\sqrt{2k}/2$	$2\sqrt{k}$
b_{opt}	$\sqrt{k/2}$	$1 \dots \sqrt{k} \quad \sqrt{k} \dots 1$
T_{opt}	$2\sqrt{2k} - 3.5$	$2\sqrt{k} - 2.5$

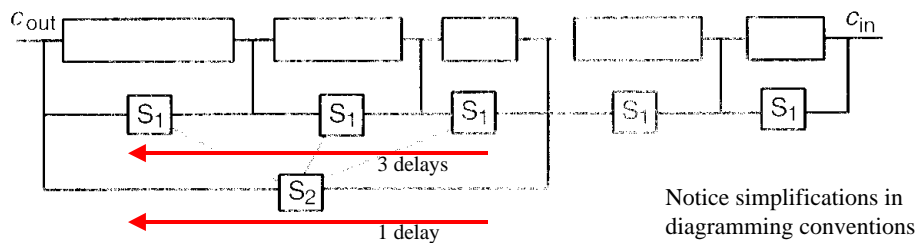
Conclusion: Variable-width is about 40% faster.

Multilevel Carry-Skip Adders

One-level carry-skip adder



Two-level carry-skip adder

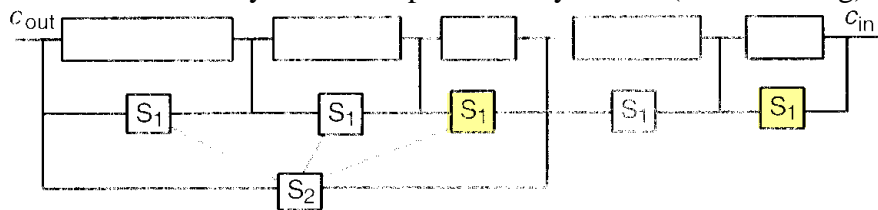


Multilevel Carry-Skip Adders

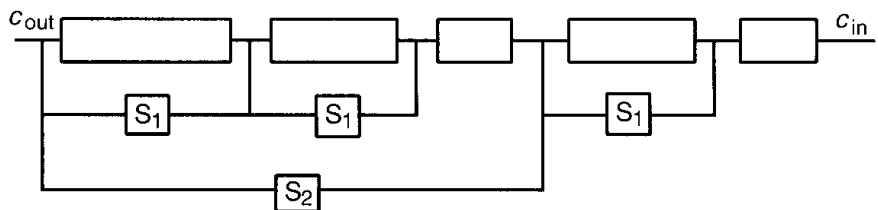
- Allow carry to skip over several level-1 skip blocks at once.
- Level-2 propagate is AND of level-1 propagates.
- Assumptions:
 - OR gate is no delay (insignificant delay)
 - Basic delay = Skip delay = Ripple delay
= Propagate Computation = Sum Computation

Simplifying the Circuit

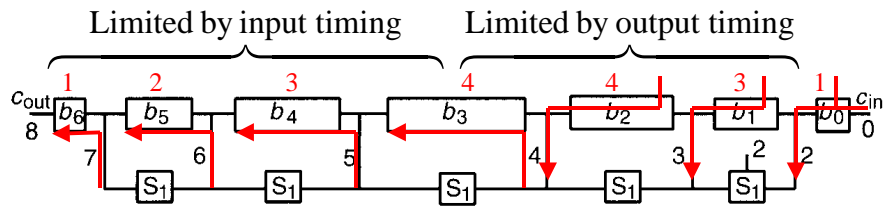
It doesn't save any time to skip short carry-chains (1-2 cells long)



optimized



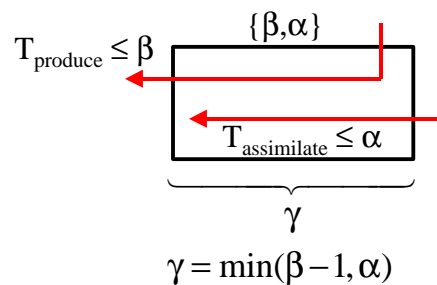
Build the Widest Single-Level Carry-Skip Adder with 8 delays max



$$Width = 1 + 3 + 4 + 4 + 3 + 2 + 1 = 18 \text{ bits}$$

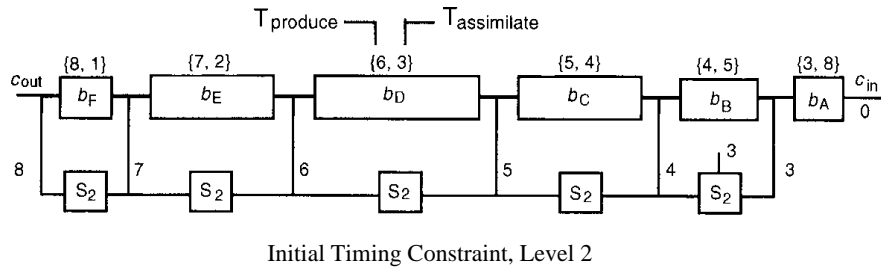
Build the Widest Two-Level Carry-Skip Adder with 8 delays max

First, we need a new notation:



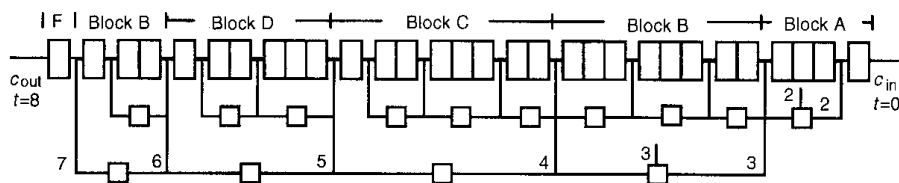
8-delay, 2-level, continued

1. Find $\{\beta, \alpha\}$ for level two



8-delay, 2-level, continued

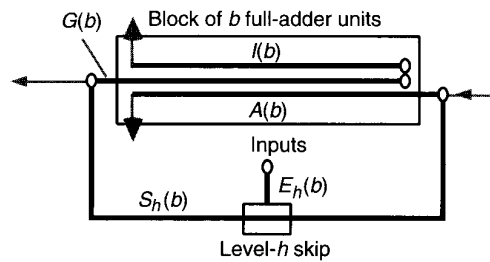
2. Given $\{\beta, \alpha\}$ for level two, derive level one



Block	$T_{produce}$	$T_{assimilate}$	Number of subblocks	Subblock widths (bits)	Block Width (bits)
A	3	8	2	1, 3	4
B	4	5	3	2, 3, 3	8
C	5	4	4	2, 3, 2, 1	8
D	6	3	3	3, 2, 1	6
E	7	2	2	2, 1	3
F	8	1	1	1	1

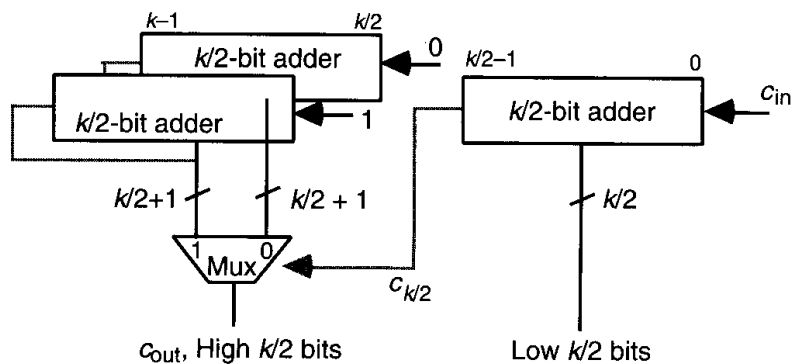
Generalization

- Chan et al. [1992] relax assumptions to include general worst-case delays:
 - $I(b)$ Internal carry-propagate delay for the block
 - $G(b)$ Carry-generate delay for the block
 - $A(b)$ Carry-assimilate delay for the block



- Used dynamic programming to obtain optimal configuration

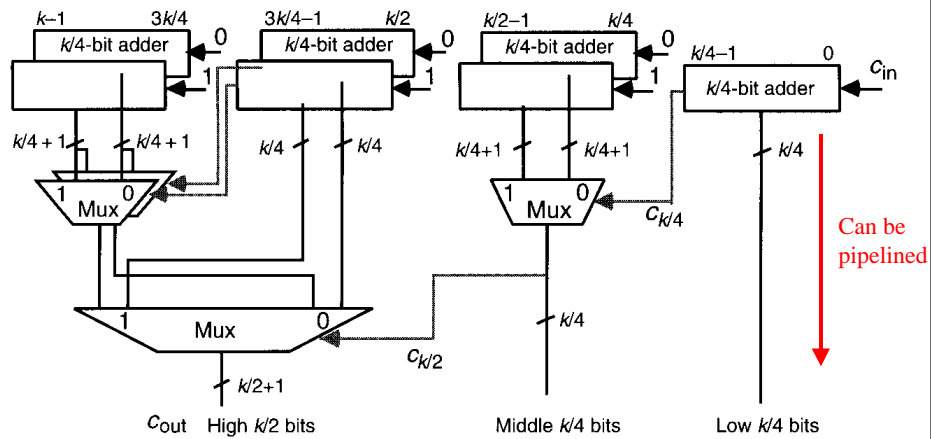
Carry-Select Adders



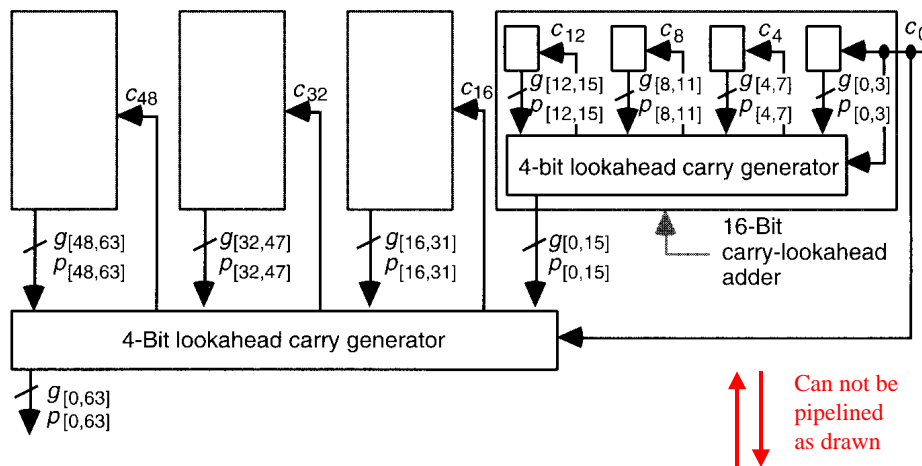
$$C_{\text{select-add}}(k) = 3C_{\text{add}}(k/2) + k/2 + 1$$

$$T_{\text{select-add}}(k) = T_{\text{add}}(k/2) + 1$$

Carry-select: Carried one step further Two-Level Carry-Select Adder



Compare to Two-Level G-P Adder



Conditional Sum Adder

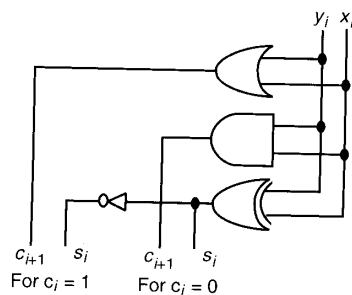
- The process that led to the two-level carry-select adder can be continued . . .
- A logarithmic time *conditional-sum adder* results if we proceed to the extreme:
 - single bit adders at the top
- A conditional-sum adder is actually a $(\log_2 k)$ -level carry-select adder

Cost and Delay of a Conditional-Sum Adder

$$C(k) \approx 2C(k/2) + k + 2 \approx k(\log_2 k + 2) + kC(1)$$

$$T(k) = T(k/2) + 1 = \log_2 k + T(1)$$

More exact analysis gives actual cost = $(k - 1)(\log_2 k + 1) + kC(1)$



Top-level block for one bit position of a conditional-sum adder

$C(1)$ and $T(1)$ are the cost and time delay of this circuit.

Conditional-Sum Example

Block width	Block carry-in	$\begin{matrix} x \\ y \end{matrix}$	Block sum and block carry-out																C_{in}	
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
1	0	$\begin{matrix} s \\ c \end{matrix}$	0	0	1	1	0	0	1	0	1	1	0	1	0	1	1	0	1	0
	1	$\begin{matrix} s \\ c \end{matrix}$	1	0	1	0	1	0	0	1	1	0	1	0	1	0	1	0	0	0
2	0	$\begin{matrix} s \\ c \end{matrix}$	0	1	1	0	1	0	1	1	0	1	0	1	0	1	1	1	0	1
	1	$\begin{matrix} s \\ c \end{matrix}$	1	0	1	1	0	0	1	0	1	0	1	0	1	0	1	0	1	0
4	0	$\begin{matrix} s \\ c \end{matrix}$	0	1	1	1	0	0	0	0	1	1	0	0	1	1	0	1	1	1
	1	$\begin{matrix} s \\ c \end{matrix}$	0	1	1	1	1	0	0	1	0	1	0	1	0	0	1	1	1	1
8	0	$\begin{matrix} s \\ c \end{matrix}$	0	1	1	1	1	0	0	0	0	1	1	0	1	0	0	1	1	1
	1	$\begin{matrix} s \\ c \end{matrix}$	0	1	1	1	1	0	0	1	0	1	1	0	0	0	1	1	1	1
16	0	$\begin{matrix} s \\ c \end{matrix}$	0	1	1	1	1	0	0	1	0	0	1	0	0	0	1	1	1	1
	1	$\begin{matrix} s \\ c \end{matrix}$																		

C_{out}

Hybrid Adder Designed

- Hybrids are obtained by combining elements of:
 - Ripple-carry adders
 - Carry-lookahead (generate-propagate) adders
 - Carry-skip adders
 - Carry-select adders
 - Conditional-sum adders
- You can obtain adders with
 - higher performance
 - greater cost-effectiveness
 - lower power consumption

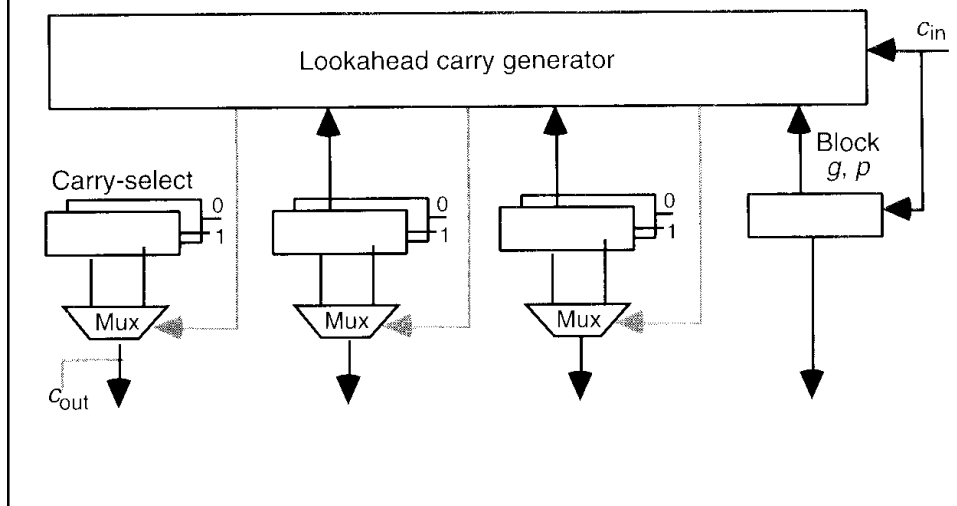
Example 1

Carry-Select / Carry-Lookahead

- One- and Two-level carry select adders are essentially hybrids, since the top level $k/2$ - or $k/4$ -bit adders can be of any type.
- Often combined with carry-lookahead adders.

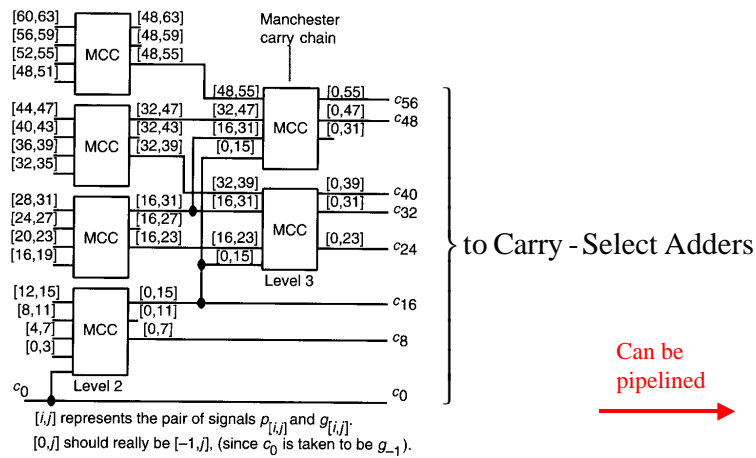
Example 2

Carry-Lookahead/Carry-Select



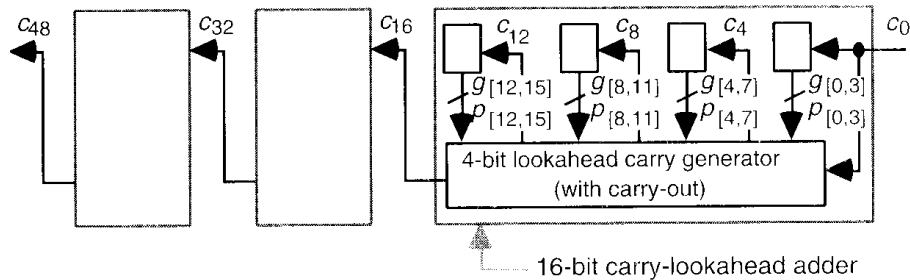
Example 3

Multilevel Carry-Lookahead/Carry-Select



Example 4

Ripple-Carry/Carry-Lookahead



Simple and modular

Example 5

Carry Lookahead/Conditional-Sum

- Reduces fan-out required to control the muxes at the lower level (a draw-back of wide conditional sum adders).
- Use carry conditional-sum addition in smaller blocks, but form inter-block carries using carry-lookahead.

Open Questions

- Application requirements may shift the balance in favor of a particular hybrid design.
- What combinations are useful for:
 - low power addition
 - addition on an FPGA

Optimizations in Fast Adders

- It is often possible to reduce the delay of adders (including hybrids) by optimizing block widths.
- The exact optimal configuration is highly technology dependent.
- Designs that minimize or regularize the interconnect may actually be more cost-effective than a design with low gate count.

Other Optimizations

- Assumption: all inputs are available at time zero.
- But, sometimes that is not true:
 - I/O arrive/depart serially, or
 - Different arrival times are associated with input digits, or
 - Different production times are associated with output digits.
- Example: Addition of partial products in a multiplier.