The Look-Ahead Adder

Lecture 22
Section C.6

Robb T. Koether

Hampden-Sydney College

Wed, Oct 23, 2019

Outline

The Carry Bits

- In the case of the ripple adder, we saw some of the operations could be done simultaneously, such as adding a_i and b_i.
- Other operations had to wait for the carry-in bit from the previous column.
- The look-ahead adder pushes this idea much further.
- It puts a premium on computing the carry-in bits for all columns as quickly as possible.

Outline

The Carry Bits

- In the case of the ripple adder, we saw that the carry-out bit may be computed as

$$ab + ac + bc,$$

  where a and b are the two bits to be added and c is the carry-in bit.
- This formula will give the carry-out bit for each column of a multi-bit adder.

The Carry Bits

The Carry Bits

$$
\begin{aligned}
c_0 &= 0, \\
c_1 &= b_0 c_0 + a_0 c_0 + a_0 b_0 = (a_0 + b_0) c_0 + a_0 b_0, \\
c_2 &= b_1 c_1 + a_1 c_1 + a_1 b_1 = (a_1 + b_1) c_1 + a_1 b_1, \\
c_3 &= b_2 c_2 + a_2 c_2 + a_2 b_2 = (a_2 + b_2) c_2 + a_2 b_2, \\
c_4 &= b_3 c_3 + a_3 c_3 + a_3 b_3 = (a_3 + b_3) c_3 + a_3 b_3, \\
&\vdots
\end{aligned}
$$

- c_i represents the carry-in bit and c_{i+1} is the carry-out bit for the i-th column.

The Carry Bits

The Carry Bits

$$
\begin{aligned}
c_1 &= b_0c_0 + a_0c_0 + a_0b_0, \\
c_2 &= b_1(b_0c_0 + a_0c_0 + a_0b_0) + a_1(b_0c_0 + a_0c_0 + a_0b_0) + a_1b_1, \\
&= b_1b_0c_0 + b_1a_0c_0 + b_1a_0b_0 + a_1b_0c_0 + a_1a_0c_0 + a_1a_0b_0 + a_1b_1, \\
&\text{etc.}
\end{aligned}
$$

- We may substitute the formula for c_1 into the expression for c_2, and so on.
- Thus, we may express each c_i in terms of the a_i's, b_i's, and c_0.
- This will quickly get out of hand.

The Generate and Propagate Bits

The Generate and Propagate Bits

$$c_{i+1} = a_i b_i + (a_i + b_i)c_i,$$
$$g_i = a_i b_i,$$
$$p_i = a_i + b_i.$$

- Instead, we will calculate the generate and propagate bits.
- In column i, a carry of 1 is *generated* if $a_i b_i = 1$.
- In column i, a carry of 1 is *propagated* from the previous column if $a_i + b_i = 1$.

The Generate and Propagate Bits

The Carry Bits

$$c_{i+1} = g_i + p_i c_i.$$

- The carry bit c_{i+1} is 1 if, in the i-th column, either
  - Generates a carry ($g_i = 1$), or
  - Propagates a carry ($p_i = 1$) from the previous column ($c_i = 1$), or
  - Both.

The Generate and Propagate Bits

The Carry Bits

$$
\begin{aligned}
c_0 &= c_0, \\
c_1 &= g_0 + p_0 c_0, \\
c_2 &= g_1 + p_1 c_1 \\
&= g_1 + p_1 (g_0 + p_0 c_0) \\
&= g_1 + p_1 g_0 + p_1 p_0 c_0, \\
c_3 &= g_2 + p_2 c_2 \\
&= g_2 + p_2 (g_1 + p_1 g_0 + p_1 p_0 c_0) \\
&= g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0,
\end{aligned}
$$

- We now have more compact expressions for c_i in terms of c_0.

The Generate and Propagate Bits

The Carry Bits

$$\begin{aligned} c_4 &= g_3 + p_3 c_3, \\ &= g_3 + p_3(g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0) \\ &= g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0, \\ &\text{etc.} \end{aligned}$$

- We now have more compact expressions for c_i in terms of c_0.

The Generate and Propagate Bits

- Using the logic of the generate and propagate bits, we first compute $g_i = a_i b_i$ and $p_i = a_i + b_i$ for all columns simultaneously.
- We can see that c_1 through c_4 can be calculated using only 5 more levels of logic.

A Four-bit Adder

1				
p_0 g_0 p_1 g_1 p_2 g_2 p_3 g_3				

- Using the logic of the generate and propagate bits, we first compute $a_i b_i$ and $a_i + b_i$ for all positions simultaneously.
- How many levels of logic are required to compute s_0, s_1, s_2, s_3, and c_4?

A Four-bit Adder

1	2			
p_0	$p_0 c_0$			
g_0	$p_1 g_0$			
p_1	$p_2 g_1$			
g_1	$p_3 g_2$			
p_2	$p_2 p_1$			
g_2	$p_3 p_2$			
p_3	$\vdots$			
g_3				

- Using the logic of the generate and propagate bits, we first compute $a_i b_i$ and $a_i + b_i$ for all positions simultaneously.
- How many levels of logic are required to compute s_0, s_1, s_2, s_3, and c_4?

A Four-bit Adder

1	2	3		
p_0	$p_0 c_0$	$p_1(p_0 c_0)$		
g_0	$p_1 g_0$	$p_2(p_1 g_0)$		
p_1	$p_2 g_1$	$p_3(p_2 g_1)$		
g_1	$p_3 g_2$	$g_1 + (p_1 g_0)$		
p_2	$p_2 p_1$	$g_2 + (p_2 g_1)$		
g_2	$p_3 p_2$	$g_3 + (p_3 g_2)$		
p_3	$\vdots$	$(p_2 p_1)(p_0 c_0)$		
g_3		$\vdots$		

- Using the logic of the generate and propagate bits, we first compute $a_i b_i$ and $a_i + b_i$ for all positions simultaneously.
- How many levels of logic are required to compute s_0, s_1, s_2, s_3, and c_4?

A Four-bit Adder

1	2	3	4	
p_0	$p_0 c_0$	$p_1(p_0 c_0)$	$p_3(p_2 p_1 p_0 c_0)$	
g_0	$p_1 g_0$	$p_2(p_1 g_0)$	$(g_1 + p_1 g_0) + (p_1 p_0 c_0)$	
p_1	$p_2 g_1$	$p_3(p_2 g_1)$	$(g_2 + p_2 g_1) + (p_2 p_1 g_0)$	
g_1	$p_3 g_2$	$g_1 + (p_1 g_0)$	$(g_3 + p_3 g_2) + (p_3 p_2 g_1)$	
p_2	$p_2 p_1$	$g_2 + (p_2 g_1)$	$\vdots$	
g_2	$p_3 p_2$	$g_3 + (p_3 g_2)$		
p_3	$\vdots$	$(p_2 p_1)(p_0 c_0)$		
g_3		$\vdots$		

- Using the logic of the generate and propagate bits, we first compute $a_i b_i$ and $a_i + b_i$ for all positions simultaneously.
- How many levels of logic are required to compute s_0, s_1, s_2, s_3, and c_4?

A Four-bit Adder

1	2	3	4	etc.
p_0	$p_0 c_0$	$p_1(p_0 c_0)$	$p_3(p_2 p_1 p_0 c_0)$	$\ldots$
g_0	$p_1 g_0$	$p_2(p_1 g_0)$	$(g_1 + p_1 g_0) + (p_1 p_0 c_0)$	$\ldots$
p_1	$p_2 g_1$	$p_3(p_2 g_1)$	$(g_2 + p_2 g_1) + (p_2 p_1 g_0)$	$\ldots$
g_1	$p_3 g_2$	$g_1 + (p_1 g_0)$	$(g_3 + p_3 g_2) + (p_3 p_2 g_1)$	$\ldots$
p_2	$p_2 p_1$	$g_2 + (p_2 g_1)$	$\vdots$	
g_2	$p_3 p_2$	$g_3 + (p_3 g_2)$		
p_3	$\vdots$	$(p_2 p_1)(p_0 c_0)$		
g_3		$\vdots$		

- Using the logic of the generate and propagate bits, we first compute $a_i b_i$ and $a_i + b_i$ for all positions simultaneously.
- How many levels of logic are required to compute s_0, s_1, s_2, s_3, and c_4?

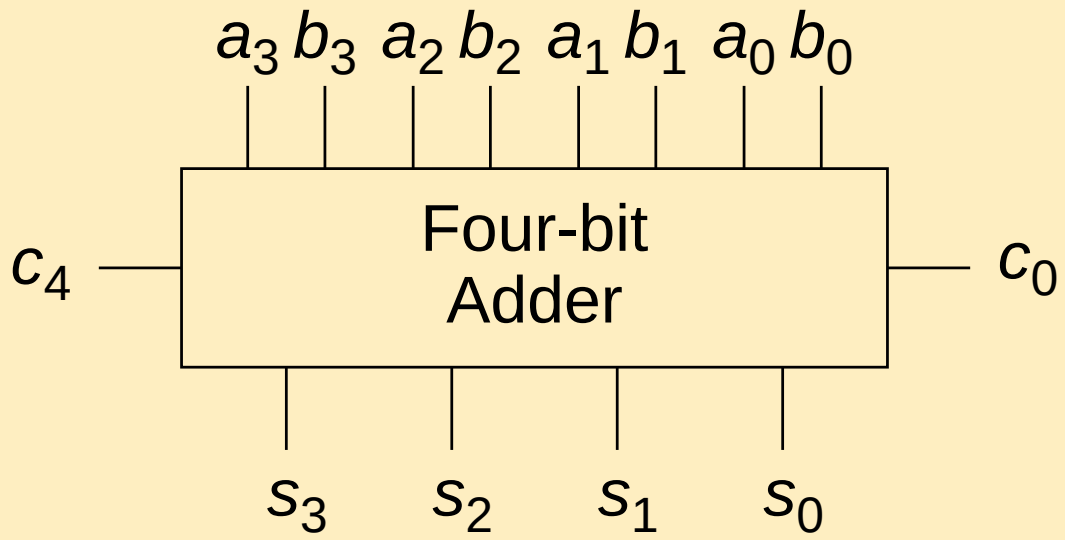
Outline

A 16-bit Adder

- It is not practical to continue in this manner through 16 bits, to say nothing of 32 bits or 64 bits.
- Instead, we view the four-bit adder as a unit.
- A 16-bit adder consists of 4 such units.
- Each unit has its own carry-in and carry-out bits.
- We apply the same logic to these 4 units.

A 16-bit Adder

A Four-bit Adder



A 16-bit Adder

Higher-level Generate and Propagate Bits

$$C_0 = c_0,$$
$$P_0 = p_3 p_2 p_1 p_0,$$
$$G_0 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0.$$

- The 4-bit adder may generate the carry bit c_4 or it may propagate it from c_0.
- We define P_0 and G_0 as shown above.
- Then we compute c_4 as

$$C_1 = G_0 + P_0 C_0.$$

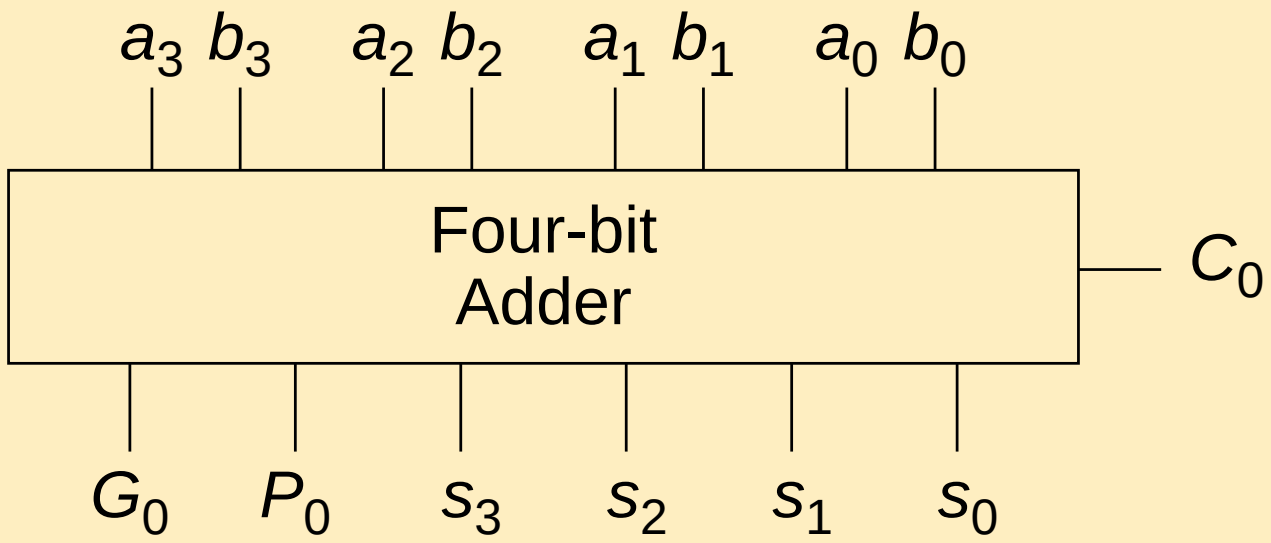
A 16-bit Adder

Higher-level Generate and Propagate Bits

$$
\begin{aligned}
C_0 &= c_0, \\
P_0 &= p_3 p_2 p_1 p_0, \\
G_0 &= g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0, \\
C_1 &= c_4 \\
&= g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0, \\
&= G_0 + P_0 C_0.
\end{aligned}
$$

A 16-bit Adder

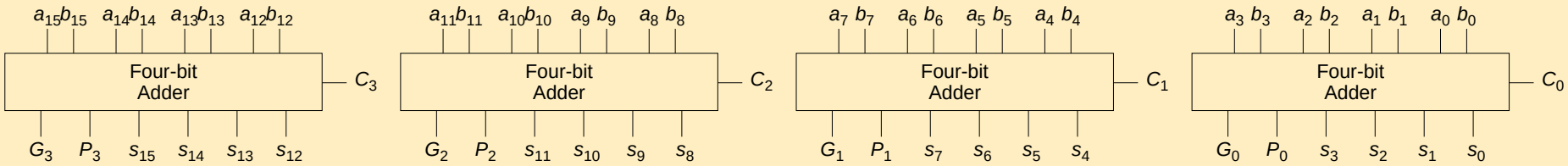
A 4-bit Adder



- Therefore, the 4-bit adder will output P_0 and G_0 instead of c_4.

A 16-bit Adder

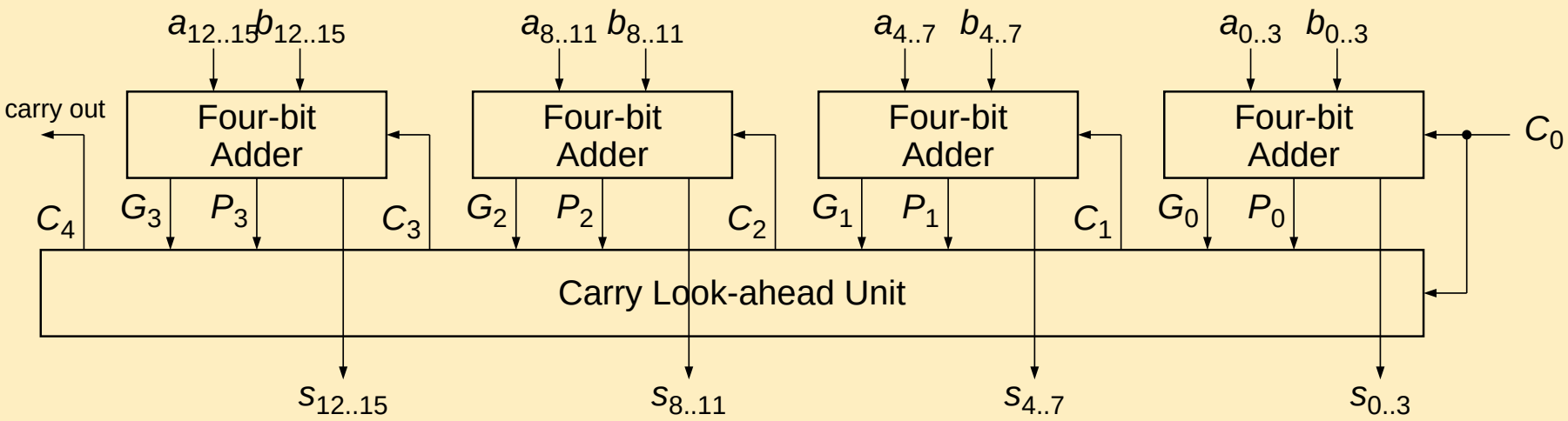
Four 4-bit Adders



- Join together four such units.

A 16-bit Adder

A 16-bit Adder



- Add the carry look-ahead unit to compute the carry-in to each 4-bit adder.
- The final carry C_4 is the carry-out of the 16-bit adder.

A 64-bit Adder

- How would we design a 32-bit look-ahead adder?
- How would we design a 64-bit look-ahead adder?

Outline

Assignment

Assignment

- Read Section C.6.