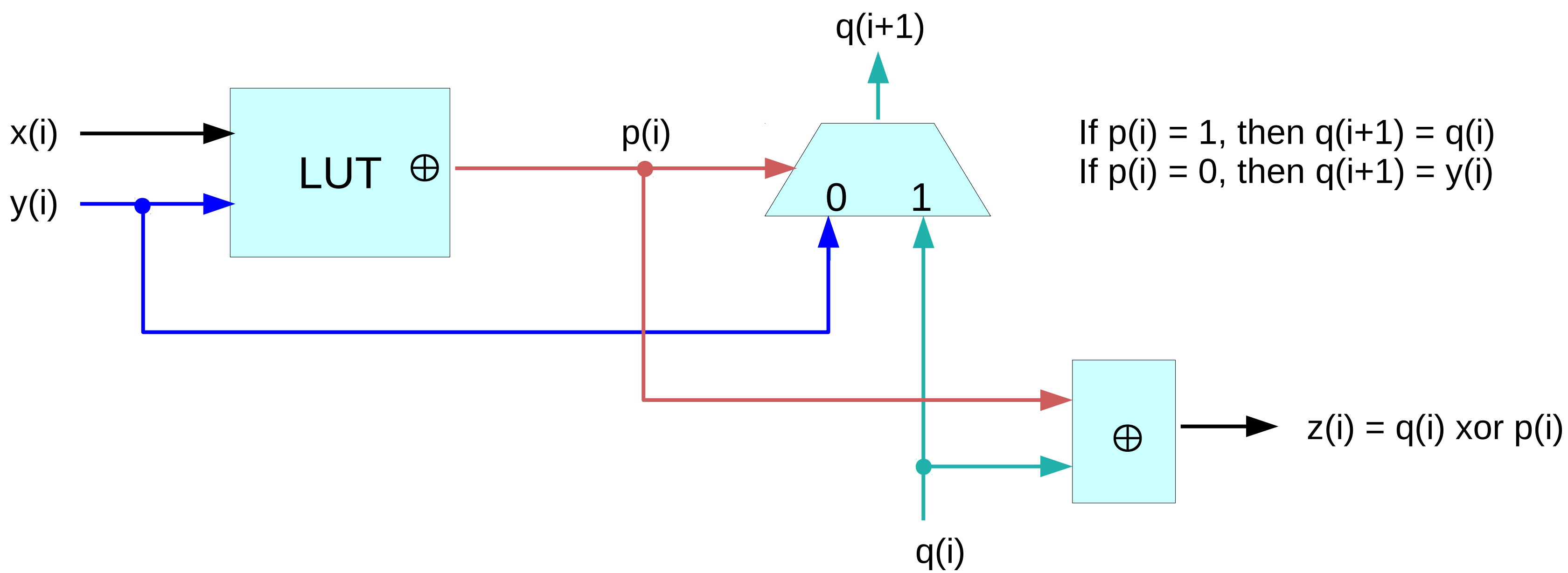
FPGA Carry Chain Adder (1A)

-
-

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

FPGA Carry Chain Cell



$$s_i = (a_i \oplus b_i) \oplus c_i = p_i \oplus c_i$$

$$c_{i+1} = (a_i \cdot b_i) + (a_i \oplus b_i) c_i = \overline{p_i} \cdot g_i + p_i \cdot c_i = \overline{p_i} \cdot a_i + p_i \cdot c_i = \overline{p_i} \cdot b_i + p_i \cdot c_i$$

when $\overline{p}_i = 1$, *then* $a_i = b_i$

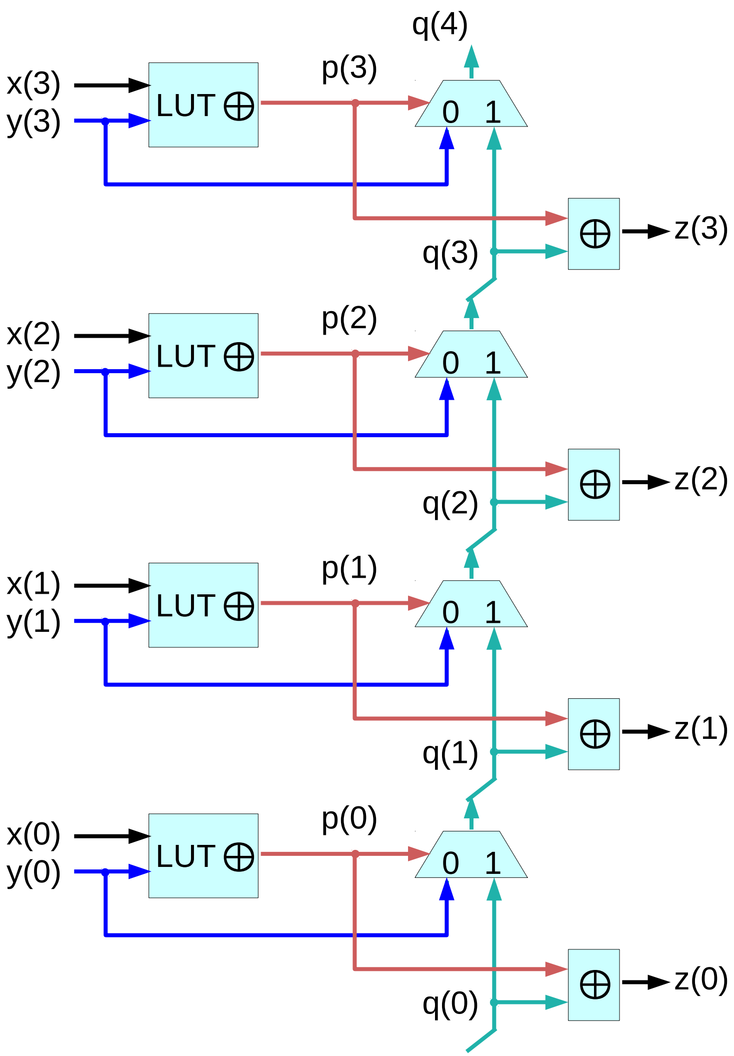
when $g_i = 1$, *then* $a_i = b_i = 1$

p(i)	0	1
0	0	1
1	1	0

g(i)	0	1
0	0	0
1	0	1

Synthesis of Arithmetic Circuits: FPGA, ASIC and Ebedded Systems, J-P Deschamps et al

FPGA Carry Chain Cell



Synthesis of Arithmetic Circuits: FPGA, ASIC and Ebedded Systems, J-P Deschamps et al

FPGA Carry Chain

FPGAs generally contain dedicated computation resources
for generating fast adders

The Virtex family programmable arrays include
logic gates (**XOR**) and **multiplexers** that along with the
general purpose **lookup tables** allow one to build effective carry-chain adders

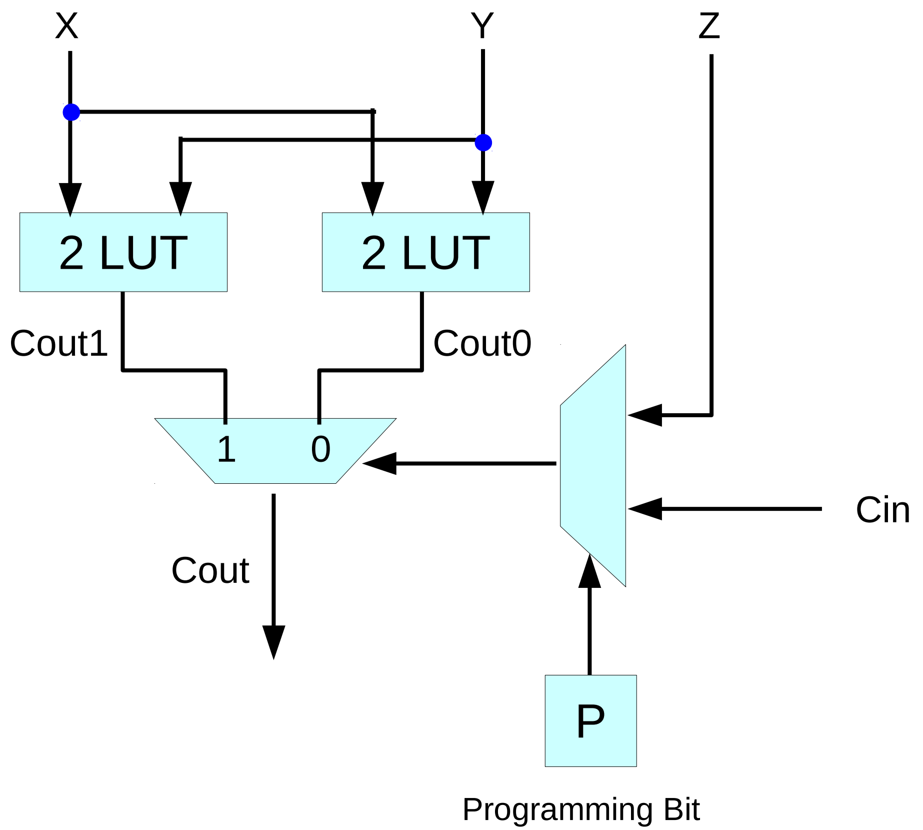
The carry chain is made up of multiplexers
belonging to adjacent configurable blocks

the lookup table is used for implementing the exclusive or function

p(i) = x(i) xor y(i)

https://en.wikipedia.org/wiki/Carry-lookahead_adder

FPGA Carry Chain Cell



Cout1, Cout2 : functions of X, Y, Cin

Cout1 = X+Y when Cin=1
Cout0 = X Y when Cin=0

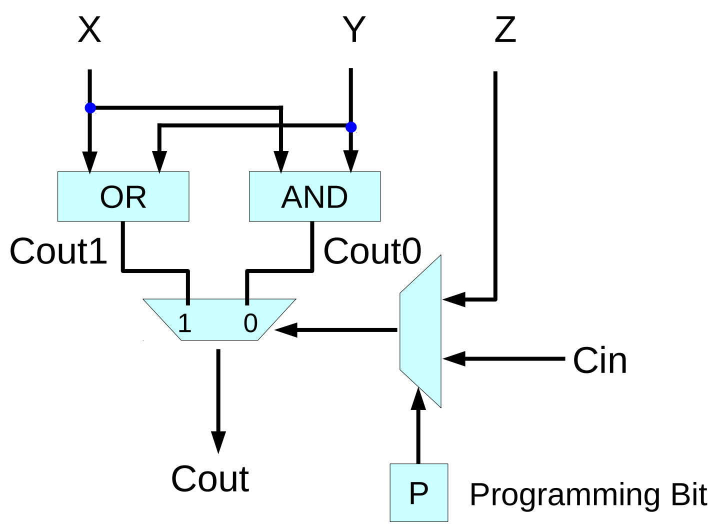
$\text{Cout} = (X + Y)\ \text{Cin} + X\ Y\ \overline{\text{Cin}}$

$\text{Cout} = P'\ \text{Cin} + G\ \overline{\text{Cin}}$ … P' = relaxed P

Cout1	Cout0	Cout	Name
0	0	0	Kill
0	1	$\overline{\text{Cin}}$	Inverse Propagate
1	0	Cin	Propagate
1	1	1	Generate

High Performance Carry Chains for FPGAs, S. Hauck, M. M. Hosler, T. W. Fry

FPGA Carry Chain Cell



		Cin	$\overline{Cin}$	
X	Y	Cout1	Cout0	
0	0	0	0	$\overline{X}\,\overline{Y}$
0	1	1	0	$\overline{X}\,Y$
1	0	1	0	$X\,\overline{Y}$
1	1	1	1	$X\,Y$

Cout : functions of X, Y, Cin

Cout(X, Y, 1) = Cout1 = X + Y
Cout(X, Y, 0) = Cout0 = X Y

Cout1 = X + Y when Cin=1
Cout0 = XY when Cin=0

Cout1 = P' Cin … P' = relaxed P
Cout0 = G $\overline{Cin}$

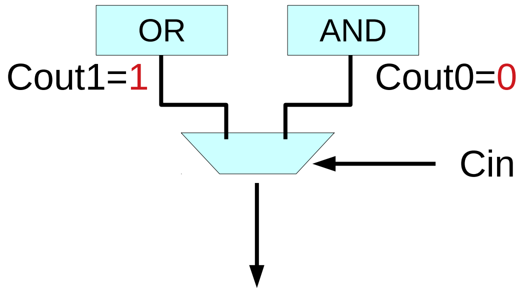
High Performance Carry Chains for FPGAs, S. Hauck, M. M. Hosler, T. W. Fry

If Cin, then Cout = ($\overline{X}$ Y + X $\overline{Y}$ + X Y)
If $\overline{Cin}$, then Cout = X Y

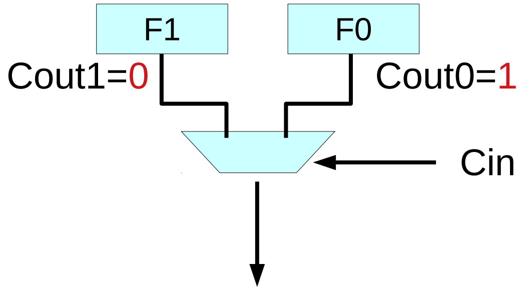
Cin (X + Y) + $\overline{Cin}$ X Y
Cin ($\overline{X}$ Y + X $\overline{Y}$ + X Y) + $\overline{Cin}$ X Y
Cin ($\overline{X}$ Y + X $\overline{Y}$) + (Cin + $\overline{Cin}$) X Y
P Cin + G

Cin (X + Y) + $\overline{Cin}$ X Y
Cin P' + $\overline{Cin}$ G … P' : relaxed P

FPGA Carry Chain Cell



Cout1=1 when Cin=1
Cout0=0 when Cin=0
Cout = Cin



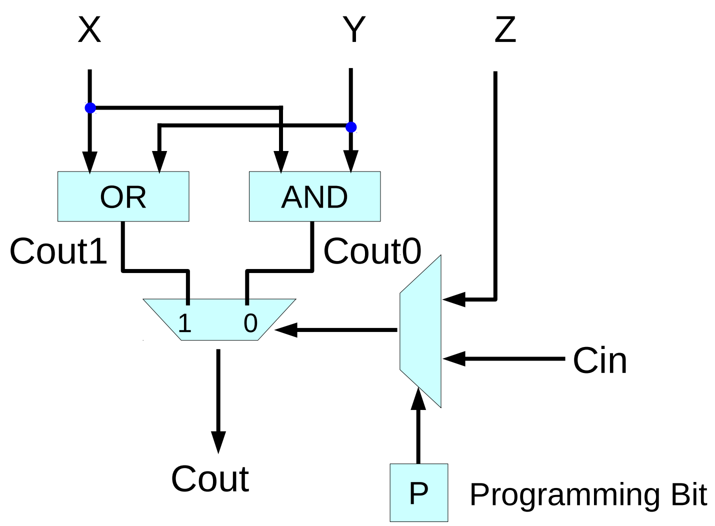
Cout1=0 when Cin=1
Cout0=1 when Cin=0
Cout = $\overline{Cin}$

Cout0	Cout1	Cout	Name
0	0	0	Kill
0	1	Cin	Propagate
1	0	$\overline{Cin}$	Inverse Propagate
1	1	1	Generate

Cout1	Cout0	Cout	Name
0	0	0	Kill
0	1	$\overline{Cin}$	Inverse Propagate
1	0	Cin	Propagate
1	1	1	Generate

High Performance Carry Chains for FPGAs, S. Hauck, M. M. Hosler, T. W. Fry

Carry Chain



X	Y	Cin: Cout1	$\overline{Cin}$: Cout0	
0	0	0	0	$\overline{X}\,\overline{Y}$
0	1	1	0	$\overline{X}\,Y$
1	0	1	0	$X\,\overline{Y}$
1	1	1	1	$X\,Y$

Cout1	Cout0	Cout	Name
0	0	0	Kill
0	1	$\overline{Cin}$	Inverse Propagate
1	0	Cin	Propagate
1	1	1	Generate

Carry Out

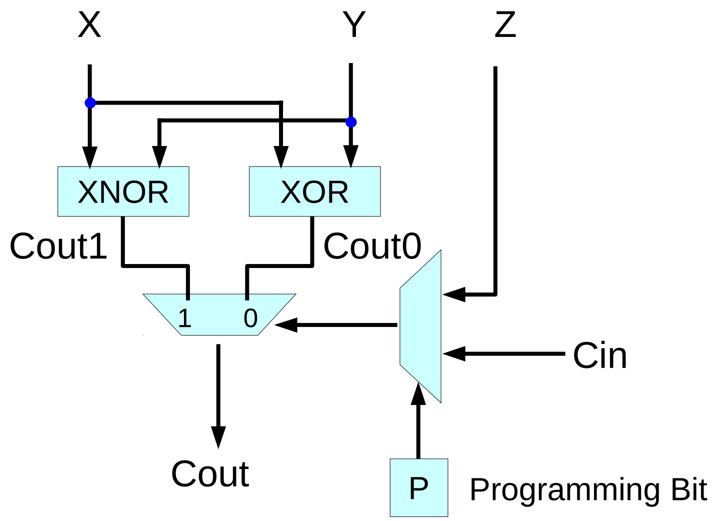
X	Y	Cin	
0	0	Cin	Cin
0	1	Cin	$\overline{Cin}$
1	0	Cin	$\overline{Cin}$
1	1	Cin	Cin

Cout1=1 when Cin=1
Cout0=0 when Cin=0
Cout = Cin propage

Cout1=0 when Cin=1
Cout0=1 when Cin=0
Cout = $\overline{Cin}$ inverse propagate

High Performance Carry Chains for FPGAs, S. Hauck, M. M. Hosler, T. W. Fry

Parity Checker



X	Y	Cout1 (Cin)	Cout0 ($\overline{Cin}$)	
0	0	1	0	$\overline{X}\ \overline{Y}$
0	1	0	1	$\overline{X}$ Y
1	0	0	1	X $\overline{Y}$
1	1	1	0	X Y

Cout1	Cout0	Cout	Name
0	0	0	Kill
0	1	$\overline{Cin}$	Inverse Propagate
1	0	Cin	Propagate
1	1	1	Generate

Computing Parity

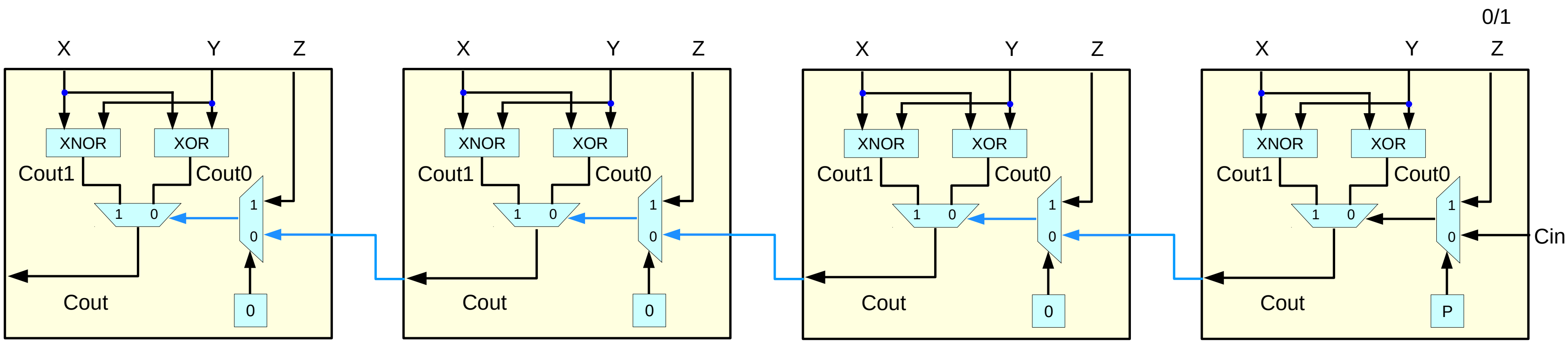
X ⊕ Y ⊕ Cin	
0 ⊕ 0 ⊕ Cin	Cin
0 ⊕ 1 ⊕ Cin	$\overline{Cin}$
1 ⊕ 0 ⊕ Cin	$\overline{Cin}$
1 ⊕ 1 ⊕ Cin	Cin

Cout1=1 when Cin=1
Cout0=0 when Cin=0
Cout = Cin propagate

Cout1=0 when Cin=1
Cout0=1 when Cin=0
Cout = $\overline{Cin}$ inverse propagate

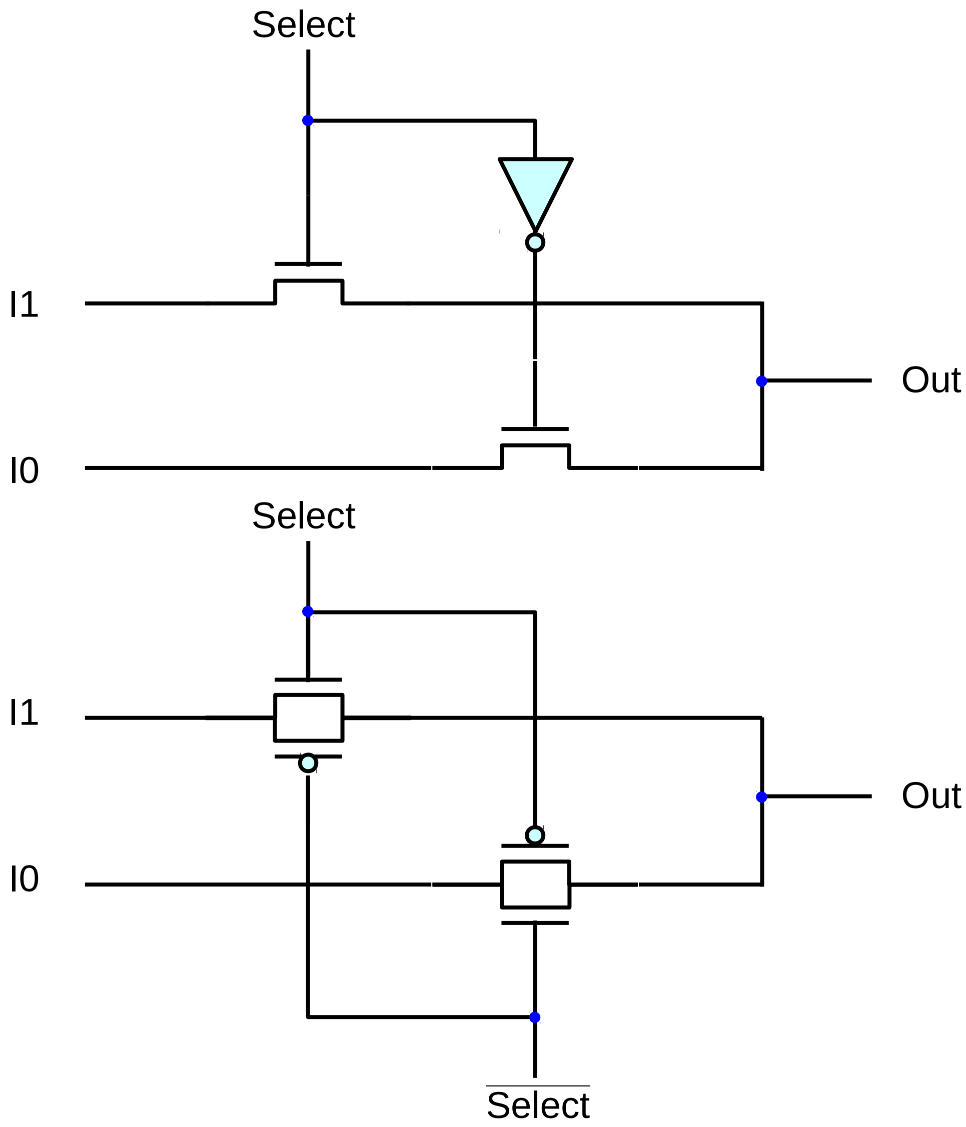
High Performance Carry Chains for FPGAs, S. Hauck, M. M. Hosler, T. W. Fry

Ripple Carry Chain



High Performance Carry Chains for FPGAs, S. Hauck, M. M. Hosler, T. W. Fry

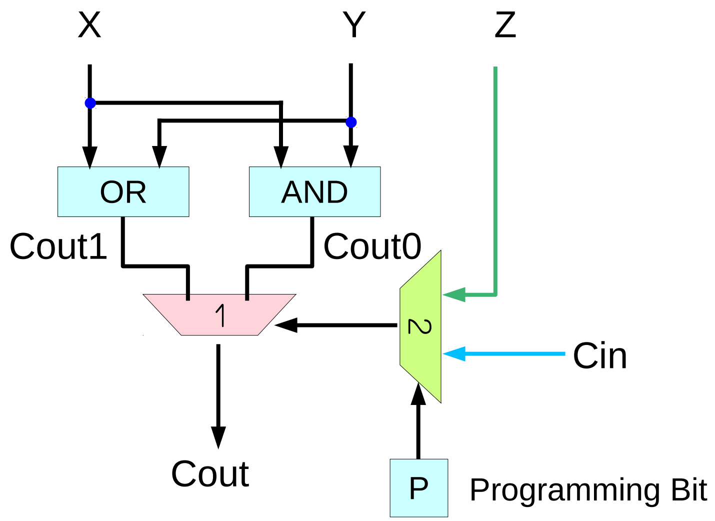
Ripple Carry Chain



P

High Performance Carry Chains for FPGAs, S. Hauck, M. M. Hosler, T. W. Fry

FPGA Carry Chain Cell

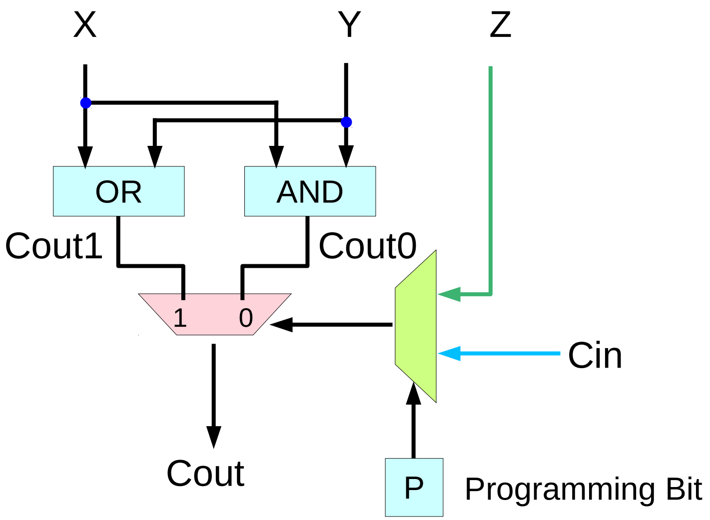


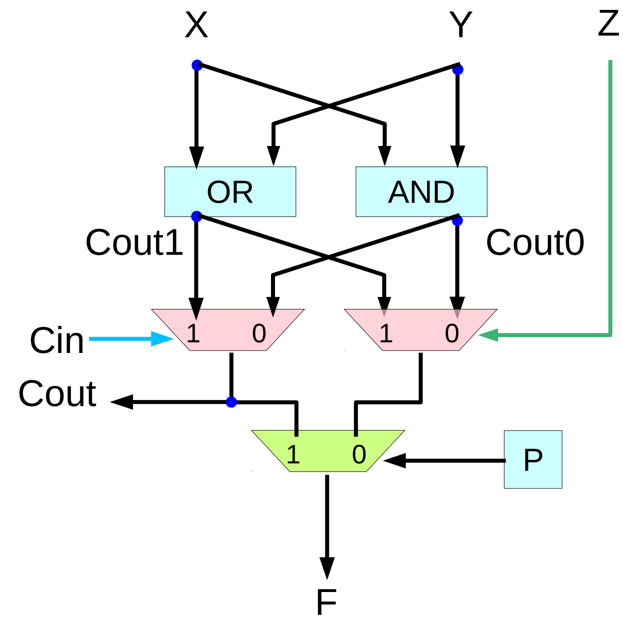
Significantly slower
Two muxes on the carry chain in each cell
Delay 1 for first cell
Delay 3 for each additional cell in the carry chain
1 delay for mux 2 and 2 delays for mux 1
Overall 2n-2 for an n-cell carry chain

The critical path comes from th 2-LUTs and not input Z
Since the delay through the 2-LUTs will be larger than
Through mux 2 in the first cell

High Performance Carry Chains for FPGAs, S. Hauck, M. M. Hosler, T. W. Fry

FPGA Carry Chain Cell





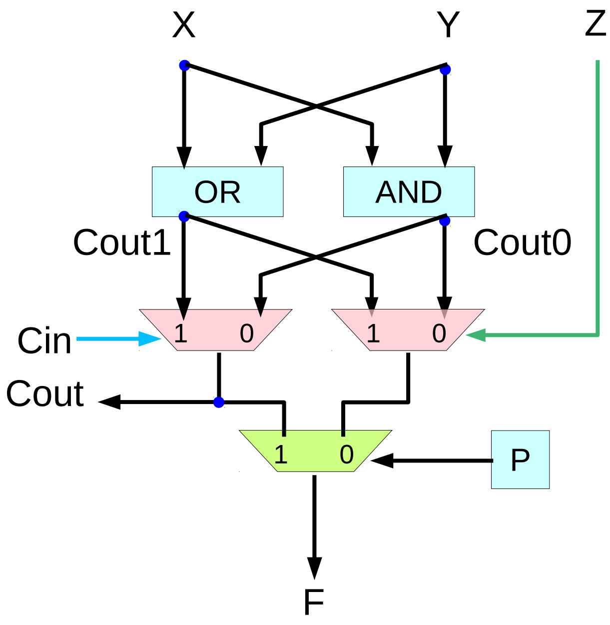
Cout1, Cout2 : functions of X, Y, Cin

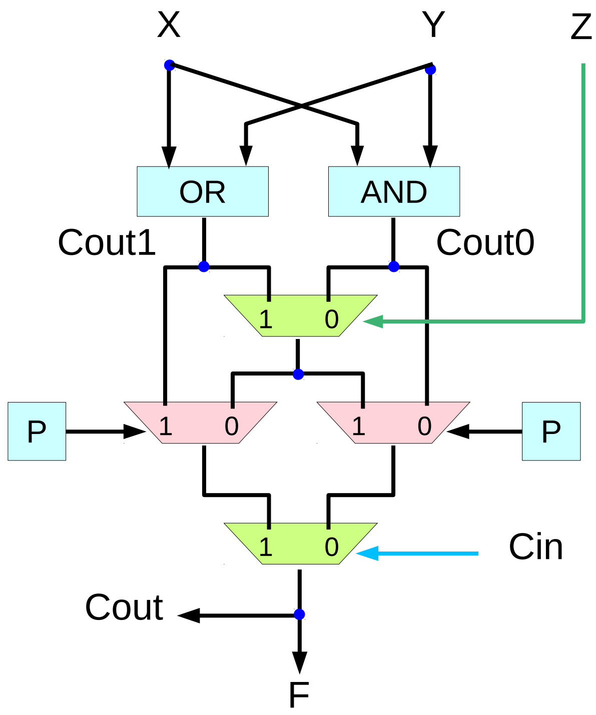
Cout1 = X+Y when Cin=1
Cout0 = X Y when Cin=0

Cout1 = Cin (X + Y)
Cout0 = $\overline{\text{Cin}}$ X Y

High Performance Carry Chains for FPGAs, S. Hauck, M. M. Hosler, T. W. Fry

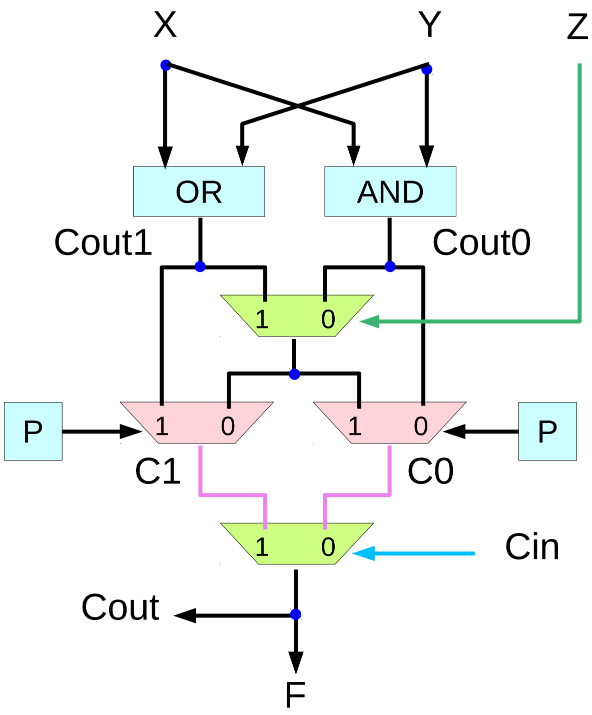
FPGA Carry Chain Cell

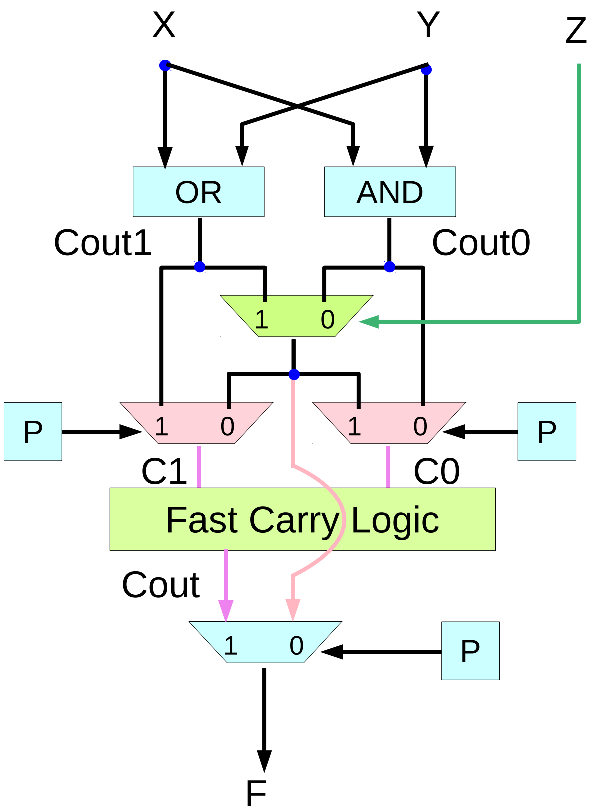




High Performance Carry Chains for FPGAs, S. Hauck, M. M. Hosler, T. W. Fry

FPGA Carry Chain Cell

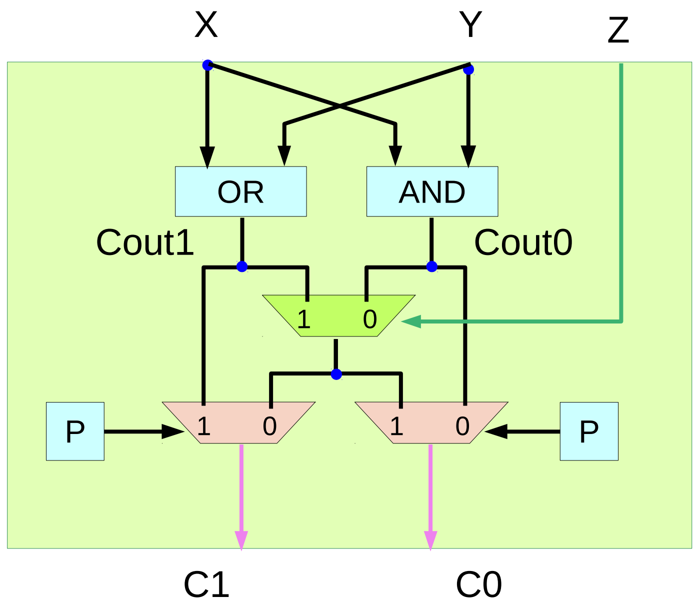




$$Cout_i = (Cout_{i-1} \cdot C1_i) + (\overline{Cout_{i-1}} \cdot C0_i)$$

High Performance Carry Chains for FPGAs, S. Hauck, M. M. Hosler, T. W. Fry

FPGA Carry Chain Cell



$$Cout_i = \left(Cout_{i-1} \cdot C1_i\right) + \left(\overline{Cout_{i-1}} \cdot C0_i\right)$$

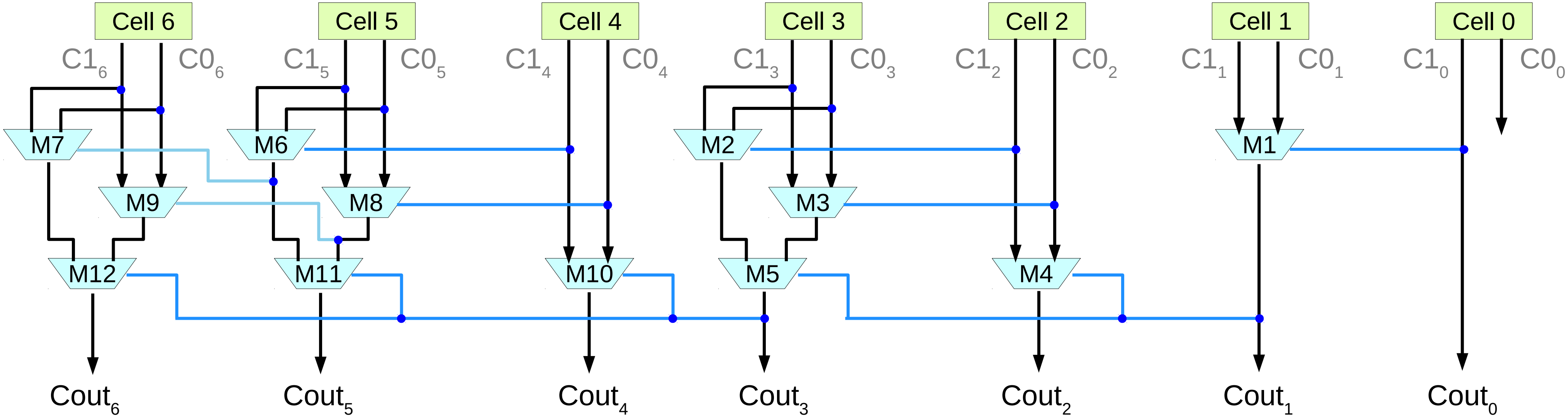
High Performance Carry Chains for FPGAs, S. Hauck, M. M. Hosler, T. W. Fry

Fast Carry Logc

Carry Select Adder
Carry Lookahead Adder
    Brent-Kung
Variable Block
Ripple Carry Adder

https://en.wikipedia.org/wiki/Carry-lookahead_adder

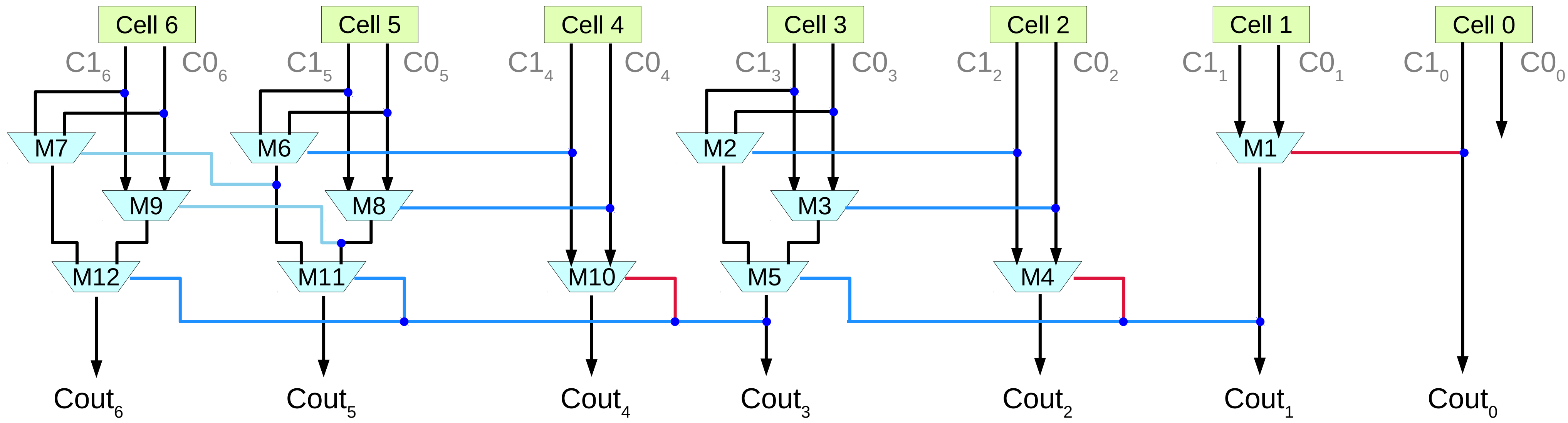
FPGA Carry Chain Cell



$$Cout_i = (Cout_{i-1} \cdot C1_i) + (\overline{Cout_{i-1}} \cdot C0_i)$$

High Performance Carry Chains for FPGAs, S. Hauck, M. M. Hosler, T. W. Fry

FPGA Carry Chain Cell



$$Cout_i = (Cout_{i-1} \cdot C1_i) + (\overline{Cout_{i-1}} \cdot C0_i)$$

$$Cout_{i+1} = (Cout_i \cdot C1_{i+1}) + (\overline{Cout_i} \cdot C0_{i+1})$$

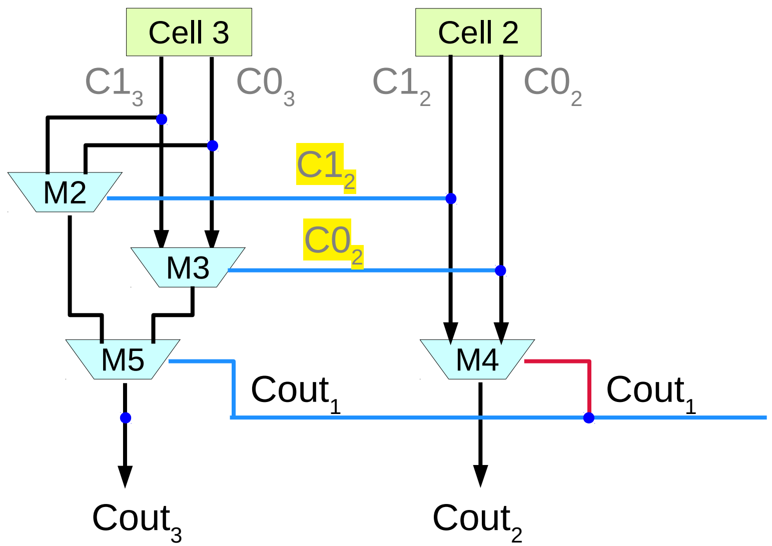
$$Cout_{i+1} = ([(Cout_{i-1} \cdot C1_i) + (\overline{Cout_{i-1}} \cdot C0_i)] \cdot C1_{i+1}) + (\overline{[(Cout_{i-1} \cdot C1_i) + (\overline{Cout_{i-1}} \cdot C0_i)]} \cdot C0_{i+1})$$

$$Cout_1 = (Cout_0 \cdot C1_1) + (\overline{Cout_0} \cdot C0_1)$$

$$Cout_1 = (C1_0 \cdot C1_1) + (\overline{C1_0} \cdot C0_1)$$

High Performance Carry Chains for FPGAs, S. Hauck, M. M. Hosler, T. W. Fry

FPGA Carry Chain Cell



$(C1_3 C1_2 + C0_3 \overline{C1_2}) Cout_1 + (C1_3 C0_2 + C0_3 \overline{C0_2}) \overline{Cout_1}$

$$Cout_i = (Cout_{i-1} \cdot C1_i) + (\overline{Cout_{i-1}} \cdot C0_i)$$

$$Cout_{i+1} = (Cout_i \cdot C1_{i+1}) + (\overline{Cout_i} \cdot C0_{i+1})$$

$$Cout_2 = (Cout_1 \cdot C1_2) + (\overline{Cout_1} \cdot C0_2)$$

$$Cout_3 = (Cout_2 \cdot C1_3) + (\overline{Cout_2} \cdot C0_3)$$

$$= (((Cout_1 \cdot C1_2) + (\overline{Cout_1} \cdot C0_2)) \cdot C1_3)$$

$$+ (\overline{((Cout_1 \cdot C1_2) + (\overline{Cout_1} \cdot C0_2))} \cdot C0_3)$$

$$(((Cout_1 \cdot C1_2) + (\overline{Cout_1} \cdot C0_2)) \cdot C1_3)$$

$$= (C1_3 C1_2 Cout_1 + C1_3 C0_2 \overline{Cout_1})$$

$$(((\overline{Cout_1 \cdot C1_2}) \cdot (\overline{\overline{Cout_1} \cdot C0_2})) \cdot C0_3)$$

$$= (((\overline{Cout_1} + \overline{C1_2}) \cdot (Cout_1 + \overline{C0_2})) \cdot C0_3)$$

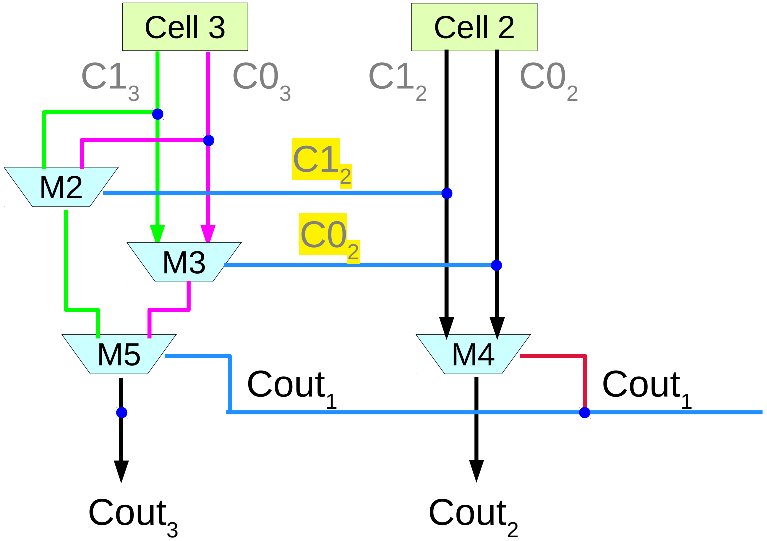
$$= (\overline{Cout_1} Cout_1 + \overline{C1_2} Cout_1 + \overline{Cout_1}\,\overline{C0_2} + \overline{C1_2}\,\overline{C0_2}) \cdot C0_3$$

$$= (\overline{C1_2} Cout_1 + \overline{C0_2}\,\overline{Cout_1}) \cdot C0_3$$

$$= (C0_3 \overline{C1_2} Cout_1 + C0_3 \overline{C0_2}\,\overline{Cout_1})$$

High Performance Carry Chains for FPGAs, S. Hauck, M. M. Hosler, T. W. Fry

FPGA Carry Chain Cell



$$= \left(\overline{Cout_1}\,Cout_1 + \overline{C1_2}\,Cout_1 + \overline{Cout_1}\,\overline{C0_2} + \overline{C1_2}\,\overline{C0_2}\right) \cdot C0_3$$

$$= \left(\overline{C1_2}\,Cout_1 + \overline{C0_2}\,\overline{Cout_1}\right) \cdot C0_3$$

$$= \left(C0_3\,\overline{C1_2}\,Cout_1 + C0_3\,\overline{C0_2}\,\overline{Cout_1}\right)$$

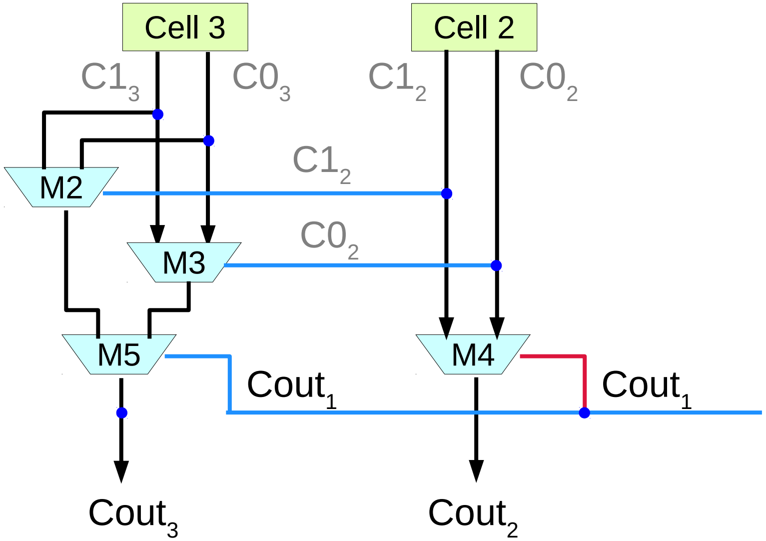
$$(C1_3\,C1_2 + C0_3\overline{C1_2})Cout_1 + (C1_3\,C0_2 + C0_3\overline{C0_2})\overline{Cout_1}$$

High Performance Carry Chains for FPGAs, S. Hauck, M. M. Hosler, T. W. Fry

FPGA Carry Chain Cell

$C1_3\,C1_2 + C0_3\,\overline{C1_2}$

$C1_3\,C0_2 + C0_3\,\overline{C0_2}$



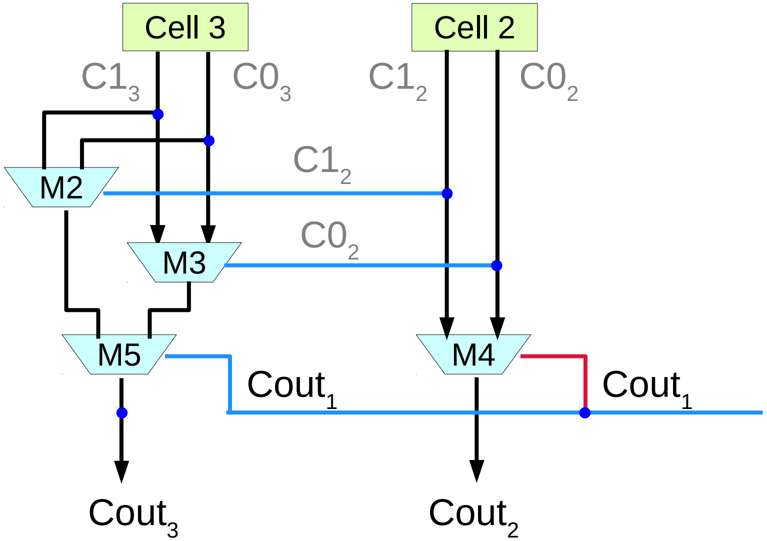
$(C1_3\,C1_2 + C0_3\,\overline{C1_2})Cout_1 + (C1_3\,C0_2 + C0_3\,\overline{C0_2})\overline{Cout_1}$

$$= C1_3 \cdot \left(C1_2 Cout_1 + C0_2 \overline{Cout_1}\right)$$

$$+ C0_3 \cdot \left(\overline{C1_2} Cout_1 + \overline{C0_2}\,\overline{Cout_1}\right)$$

High Performance Carry Chains for FPGAs, S. Hauck, M. M. Hosler, T. W. Fry

FPGA Carry Chain Cell



$$Cout_i = \left(Cout_{i-1} \cdot C1_i\right) + \left(\overline{Cout_{i-1}} \cdot C0_i\right)$$

$$Cout_{i+1} = \left(Cout_i \cdot C1_{i+1}\right) + \left(\overline{Cout_i} \cdot C0_{i+1}\right)$$

$$Cout_{i+1} = \left(\left[\left(Cout_{i-1} \cdot C1_i\right) + \left(\overline{Cout_{i-1}} \cdot C0_i\right)\right] \cdot C1_{i+1}\right)$$

$$+ \left(\overline{\left[\left(Cout_{i-1} \cdot C1_i\right) + \left(\overline{Cout_{i-1}} \cdot C0_i\right)\right]} \cdot C0_{i+1}\right)$$

High Performance Carry Chains for FPGAs, S. Hauck, M. M. Hosler, T. W. Fry

References

[1] http://en.wikipedia.org/
[2] J-P Deschamps,et. al., “Sunthesis of Arithmetic Circuits”, 2006