

CSE140 Final Review

Xinyuan Wang

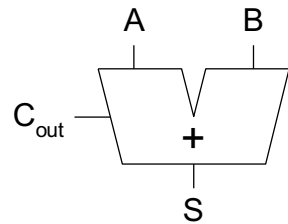
06/06/2019

Contents

- Serial Adder/Subtractor
- Counter
- Cascade counter
- Floating number

1-Bit Adders/Subtractors

Half Adder

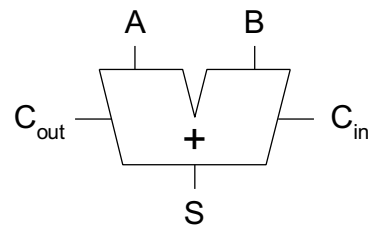


A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$S = A \oplus B$$

$$C_{out} = AB$$

Full Adder

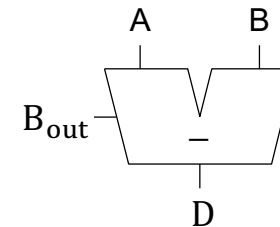


C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = AB + AC_{in} + BC_{in}$$

Half Subtractor

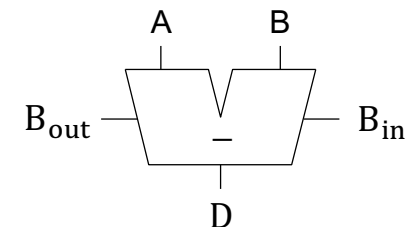


A	B	B_{out}	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

$$D = A \oplus B$$

$$B_{out} = A'B$$

Full Subtractor



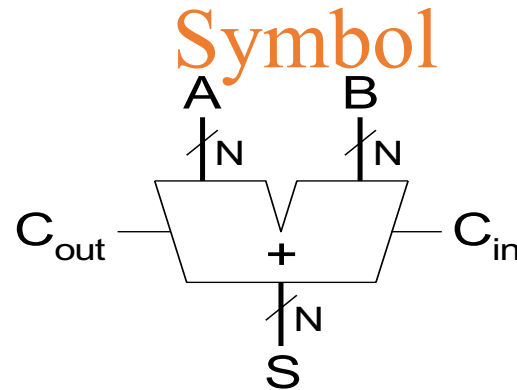
B_{in}	A	B	B_{out}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	1
0	1	1	0	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1

$$D = A \oplus B \oplus B_{in}$$

$$B_{out} = A'B + A'B_{in} + BB_{in}$$

Adder

- Several types of carry propagate adders (CPAs) are:
 - Ripple-carry adders (slow)
 - Carry-lookahead adders (fast)
 - Prefix adders (faster)
- Carry-lookahead and prefix adders are faster for large adders but require more hardware.



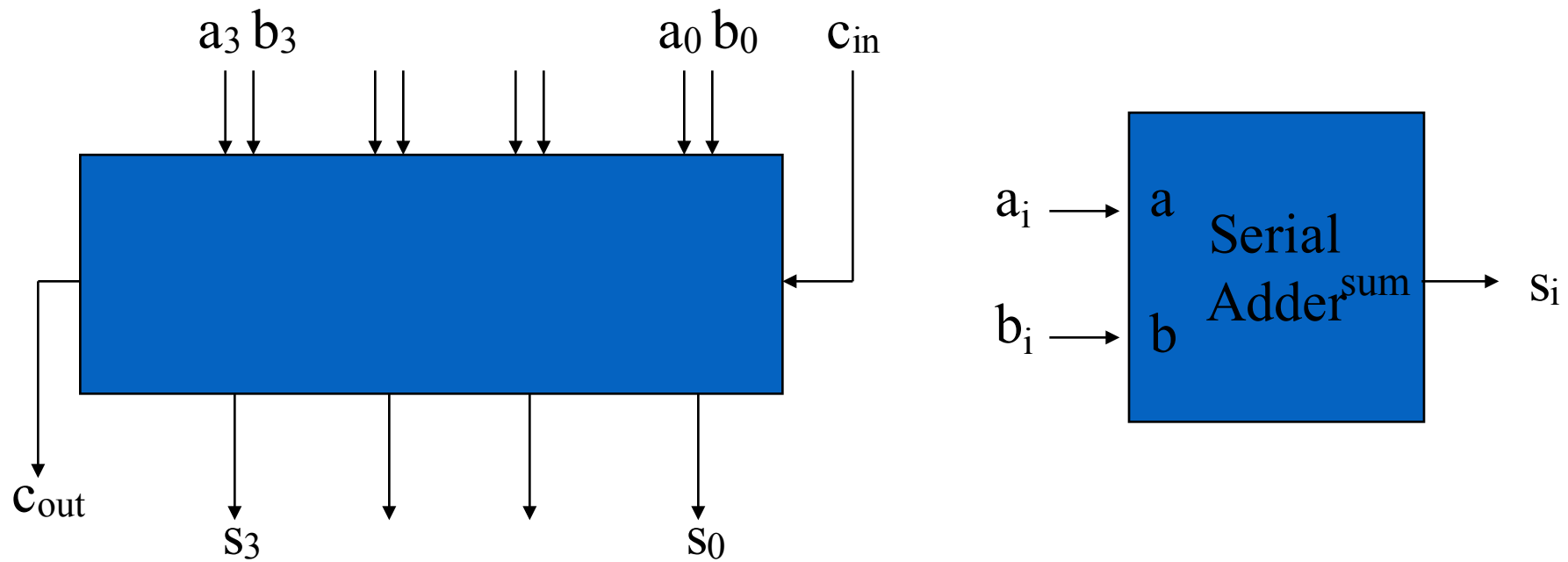
Motivation for Serial Adders and Multipliers

- Tradeoff of silicon area and system performance
 - Perform process in a series of time
- Utilization of FPGA architecture
 - Slice operation bitwise
- Metrics of Cost, Speed, and Power
- Ad: Cheaper hardware, Fit for FPGA architecture, Pipelining for excellent throughput
- Dis: Longer latency

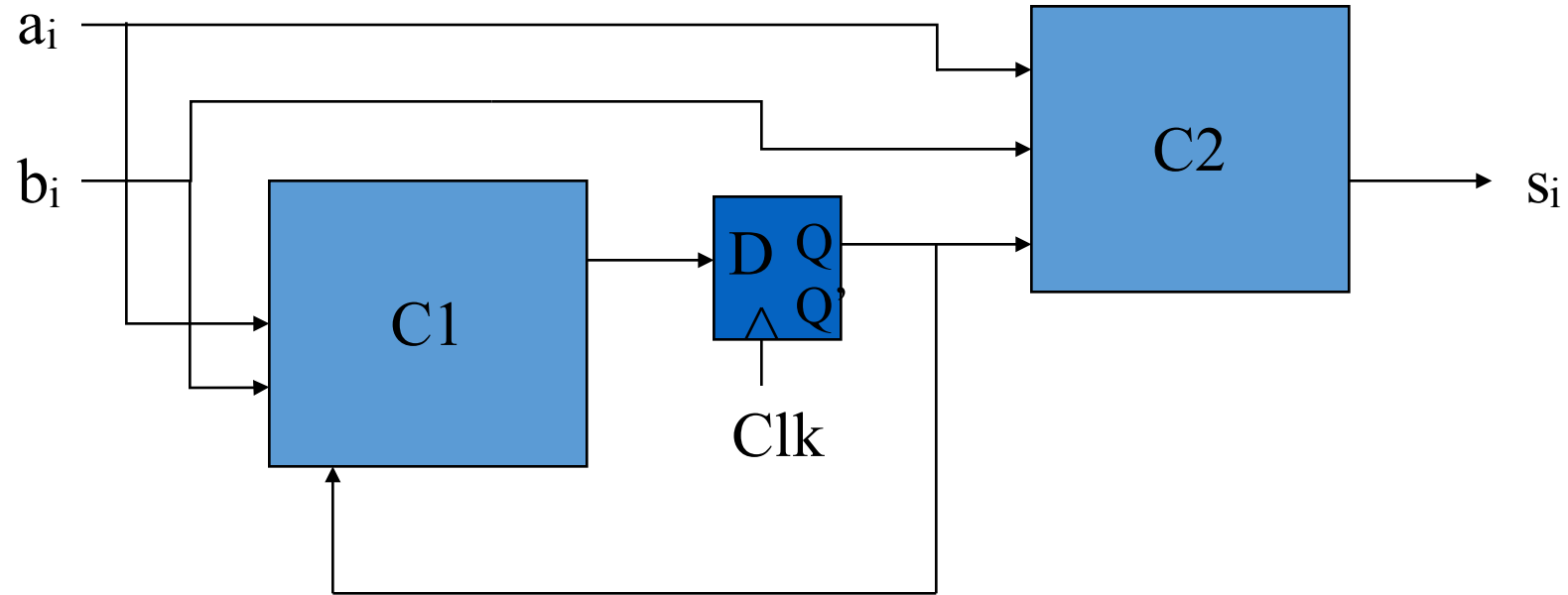
Serial Adder: Perform serial bit-addition

At time i , read a_i and b_i . Produce s_i and c_{i+1}

Internal state stores c_i . Carry bit c_0 is set as c_{in}



Serial Adder using D F-F



Feed a_i and b_i and generate s_i at time i .
Where is c_i and c_{i+1} ?

Serial Adder using a D Flip-Flop

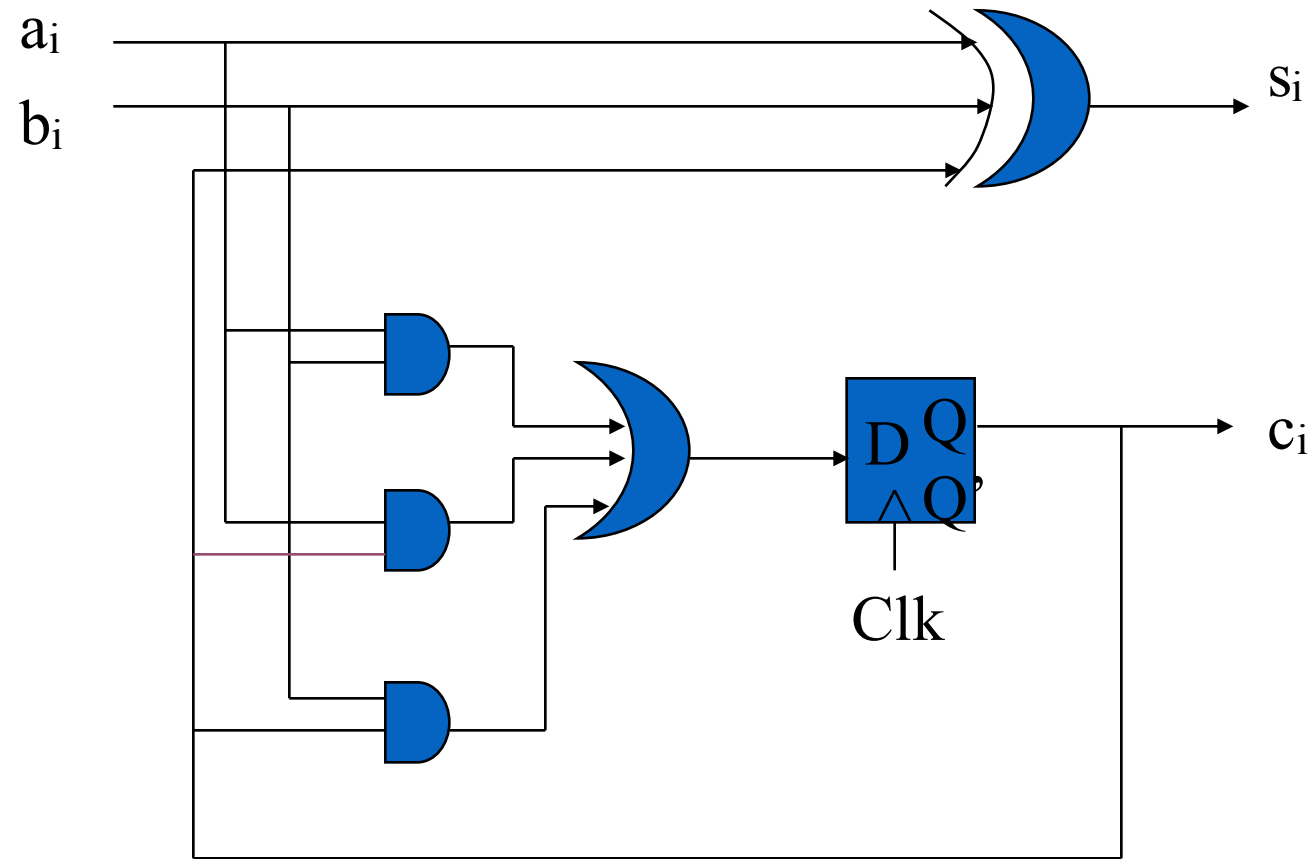
id	a_i	b_i	c_i	c_{i+1}	S_i
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	1

$$D=c_{i+1}$$

$$Q=c_i$$

Serial Adder using a D Flip-Flop

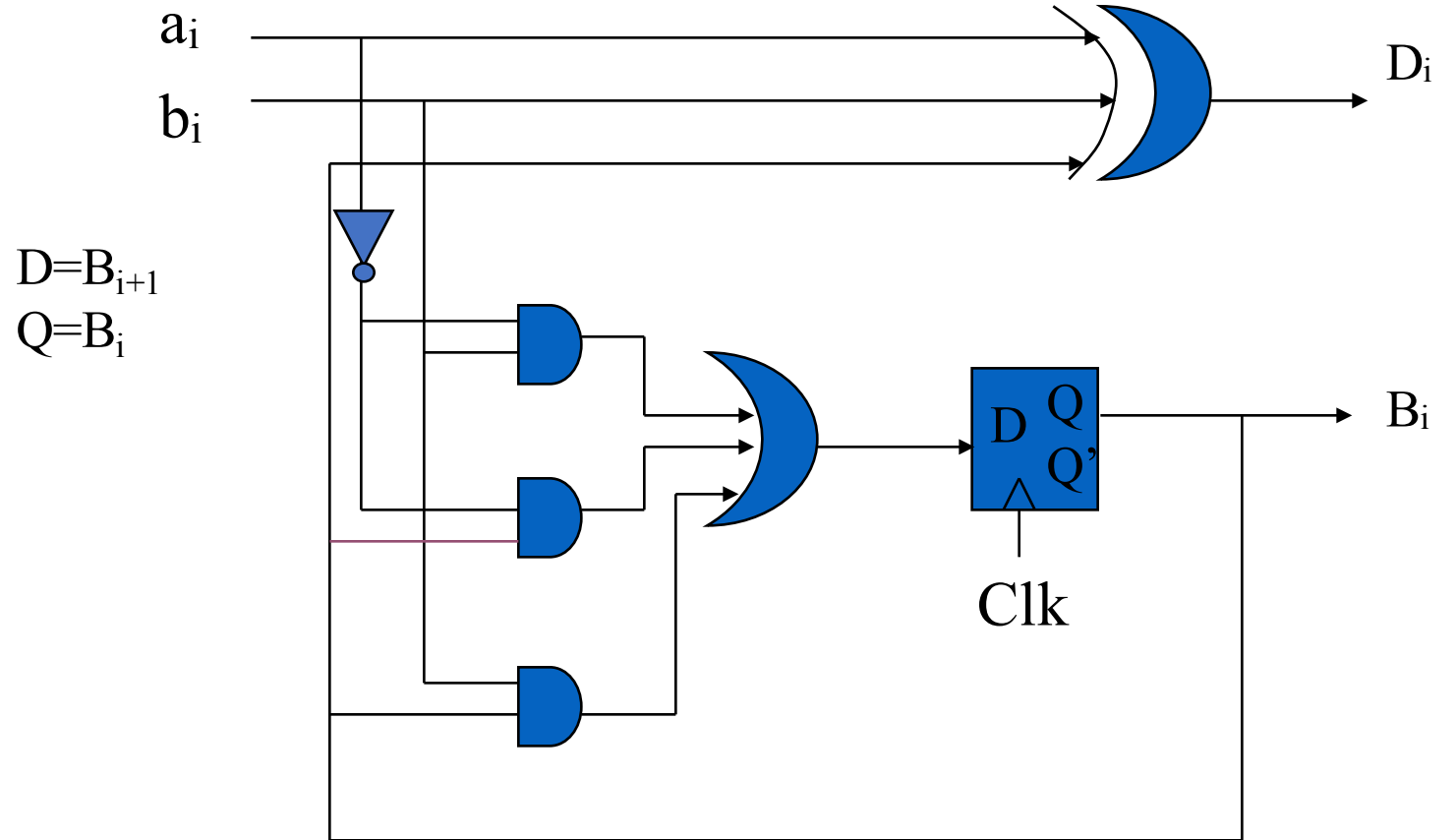
Logic Diagram



Serial Subtractor using a D Flip-Flop

Logic Diagram

id	a_i	b_i	B_i	B_{i+1}	D_i
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	1	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	1	1

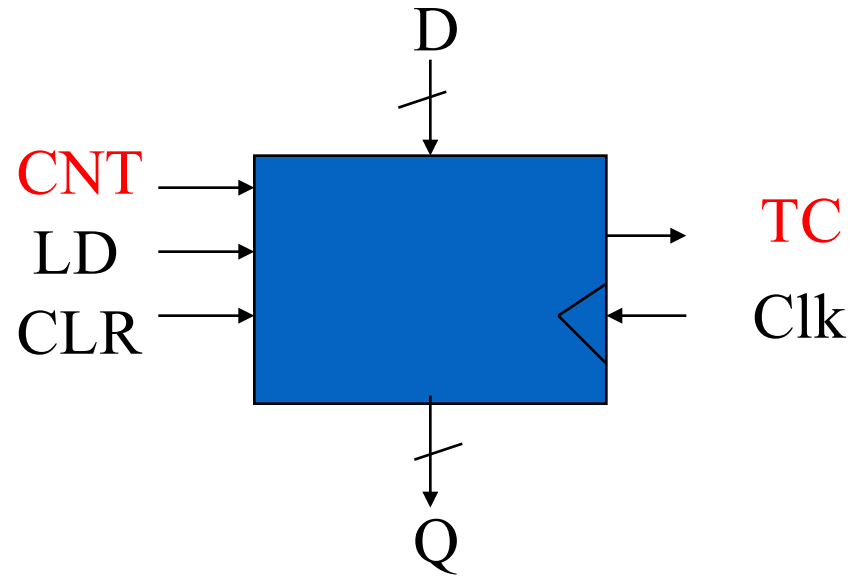


Counter

- Modulo-n Counter
- Modulo Counter ($m < n$)
- Counter (a-to-b)
- Counter of an Arbitrary Sequence
- Cascade Counter

Counter

- A modulo-n counter



$$\begin{array}{ll} Q(t+1) = (0, 0, \dots, 0) & \text{if } CLR = 1 \\ & = D \quad \text{if } LD = 1 \text{ and } CLR = 0 \\ & = (Q(t)+1) \bmod n \quad \text{if } LD = 0, CNT = 1 \text{ and } CLR = 0 \\ & = Q(t) \quad \text{if } LD = 0, CNT = 0 \text{ and } CLR = 0 \\ TC & = 1 \quad \text{if } Q(t) = n-1 \text{ and } CNT = 1 \\ & = 0 \quad \text{otherwise} \end{array}$$

Counter

- **A modulo-m counter ($m < n$)**

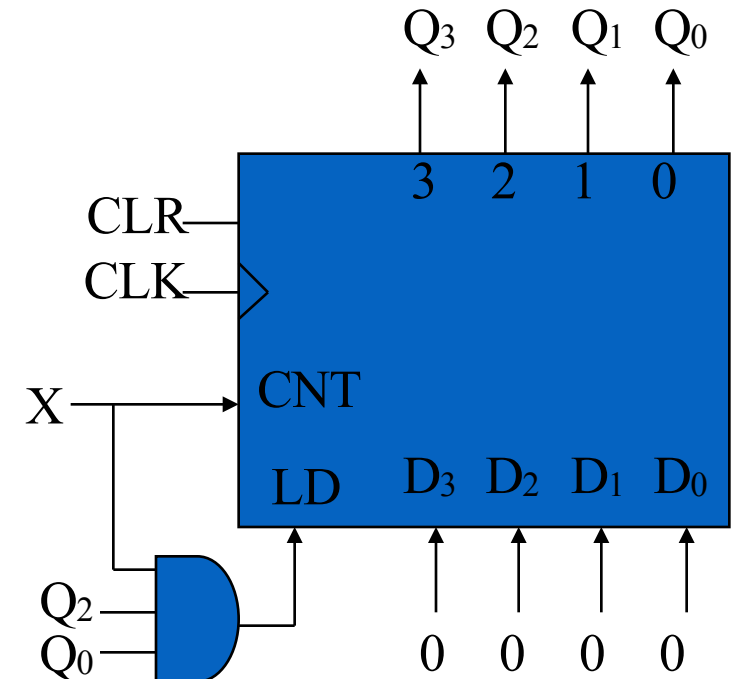
- ✓ Given a mod 16 counter
- ✓ Construct a mod-m counter ($0 < m < 16$) with AND, OR, NOT gates

m=6

if $LD = 0, X = 1$ $Q(t+1) = (Q(t) + 1) \bmod m$

- when $Q(t) < m$, works as the modulo-n counter
- when $Q(t) = m-1$, $Q(t+1)$ needs to be reset to 0

Set $LD = 1$ when $X = 1$ and $Q(t) = m-1$
 $Q(t+1) = D = 0$



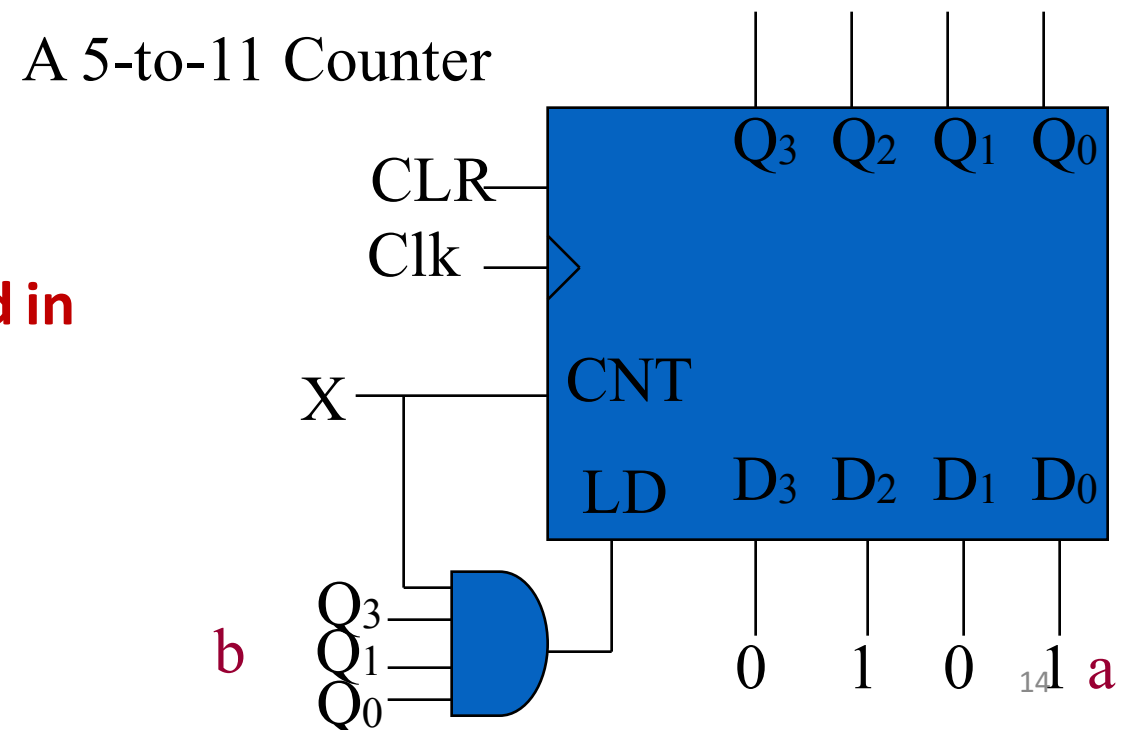
Counter

- **Design a a-to-b counter with modulo-16 counter**

- ✓ Count from a to b ($0 < a < b < 15$), a 5-to-11 counter from slide 14 of Lecture 15
 - ✓ Set $(D_3D_2D_1D_0) = a$
 - ✓ When $CNT = 1$ and $(Q_3Q_2Q_1Q_0) = b$, set **LD = 1** then $D \rightarrow Q(t + 1)$
 - ✓ Comb logic for LD only takes 1s
- ### A 5-to-11 Counter

$$(Q_3Q_2Q_1Q_0) = 1011$$

$(Q_3Q_2Q_1Q_0) = 1111$ cannot be reached in this 5-to-11 counter!



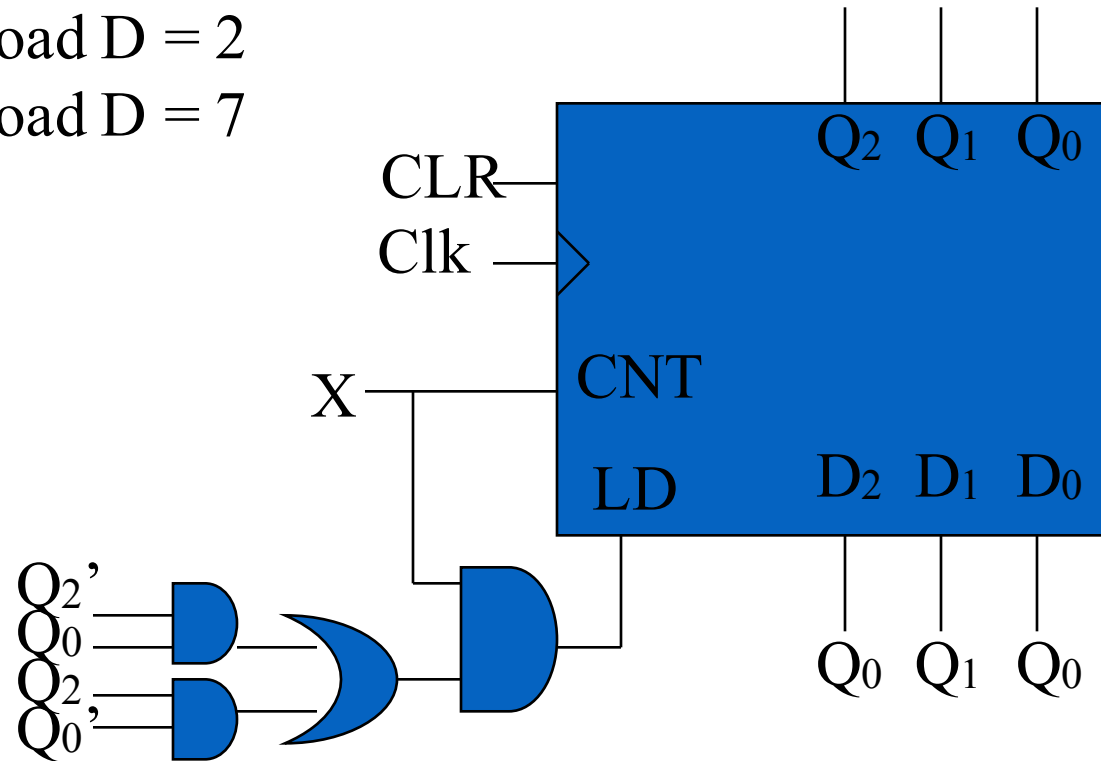
Counter of an Arbitrary Sequence

Given a mod 8 counter, construct
a counter with sequence 0 1 5 6 2 3 7

When $Q = 1$, load $D = 5$

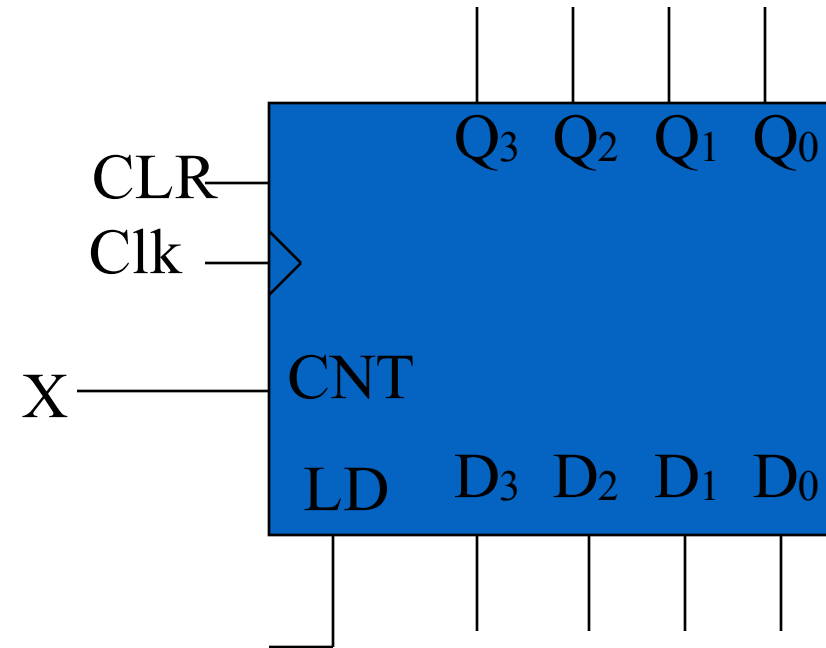
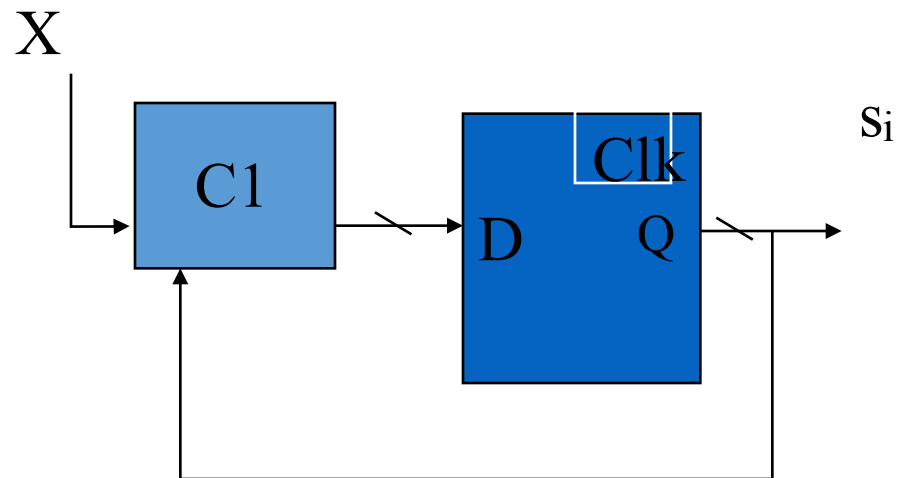
When $Q = 6$, load $D = 2$

When $Q = 3$, load $D = 7$



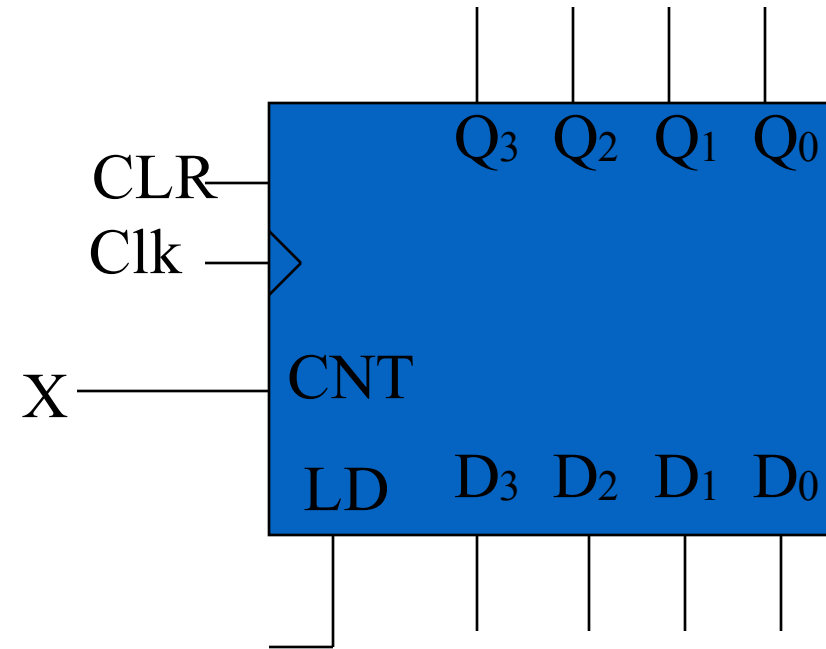
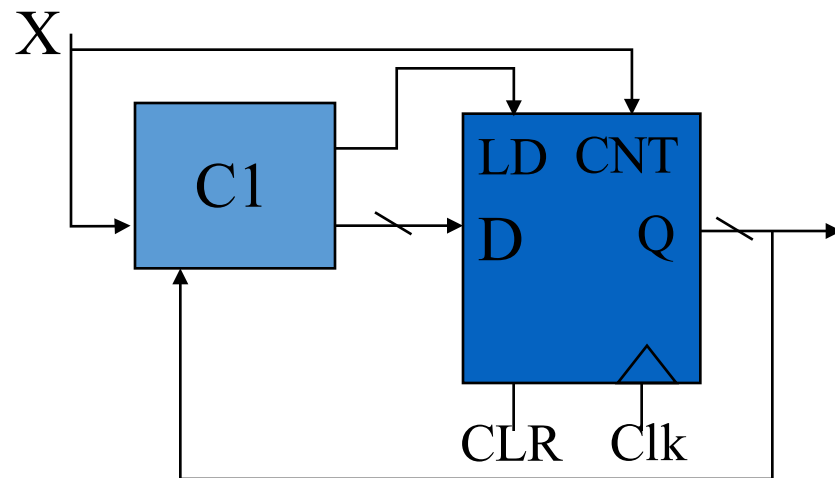
Counter

- **Design a counter of an arbitrary sequence with a modulo-16 counter**
 - ✓ an arbitrary sequence (2, 3, 5, 6, 7, 10, 11, 15, 1)
 - ✓ when $Q = 3$, load $D = 5$
 - ✓ when $Q = 7$, load $D = 10$
 - ✓ when $Q = 11$, load $D = 15$
 - ✓ when $Q = 15$, load $D = 1$



Counter

- **Design a counter of an arbitrary sequence with a modulo-16 counter**
 - ✓ an arbitrary sequence (2, 3, 5, 6, 7, 10, 11, 15, 1)
 - ✓ when $Q = 3$, load $D = 5$
 - ✓ when $Q = 7$, load $D = 10$
 - ✓ when $Q = 11$, load $D = 15$
 - ✓ when $Q = 15$, load $D = 1$



Counter

- **Design a counter of an arbitrary sequence with a modulo-16 counter**

- ✓ an arbitrary sequence (2, 3, 5, 6, 7, 10, 11, 15, 1)

- ✓ when $Q = 3$, load $D = 5$

- ✓ when $Q = 7$, load $D = 10$

- ✓ when $Q = 11$, load $D = 15$

- ✓ when $Q = 15$, load $D = 1$

- ✓ Set $LD = 1$ for load new number, otherwise $LD = 0$. If number not in the sequence, set LD as don't care.

- ✓ Set $D =$ new number when $LD = 1$, otherwise set D as don't cares.

Id	$Q_3Q_2Q_1Q_0$	LD	D_3	D_2	D_1	D_0
1	0001	0	-	-	-	-
2	0010	0	-	-	-	-
3	0011	1	0	1	0	1
5	0101	0	-	-	-	-
6	0110	0	-	-	-	-
7	0111	1	1	0	1	0
10	1010	0	-	-	-	-
11	1011	1	1	1	1	1
15	1111	1	0	0	0	1
0, 4, 8, 9, 12, 13, 14	0000 ...	-	-	-	-	-

Counter

- **Design a counter of an arbitrary sequence with a modulo-16 counter**
 - ✓ an arbitrary sequence (2, 3, 5, 6, 7, 10, 11, 15, 1)
 - ✓ Use K-maps to minimize the logic

$$LD = Q_1 Q_0 X$$

$Q_1 Q_0 \backslash Q_3 Q_2$	00	01	11	10
00	X ₀	X ₄	X ₁₂	X ₈
01	0 ₁	0 ₅	X ₁₃	X ₉
11	1 ₃	1 ₇	1 ₁₅	1 ₁₁
10	0 ₂	0 ₆	X ₁₄	0 ₁₀

$$D_3 = D_1 = Q_3' Q_2 + Q_3 Q_2'$$

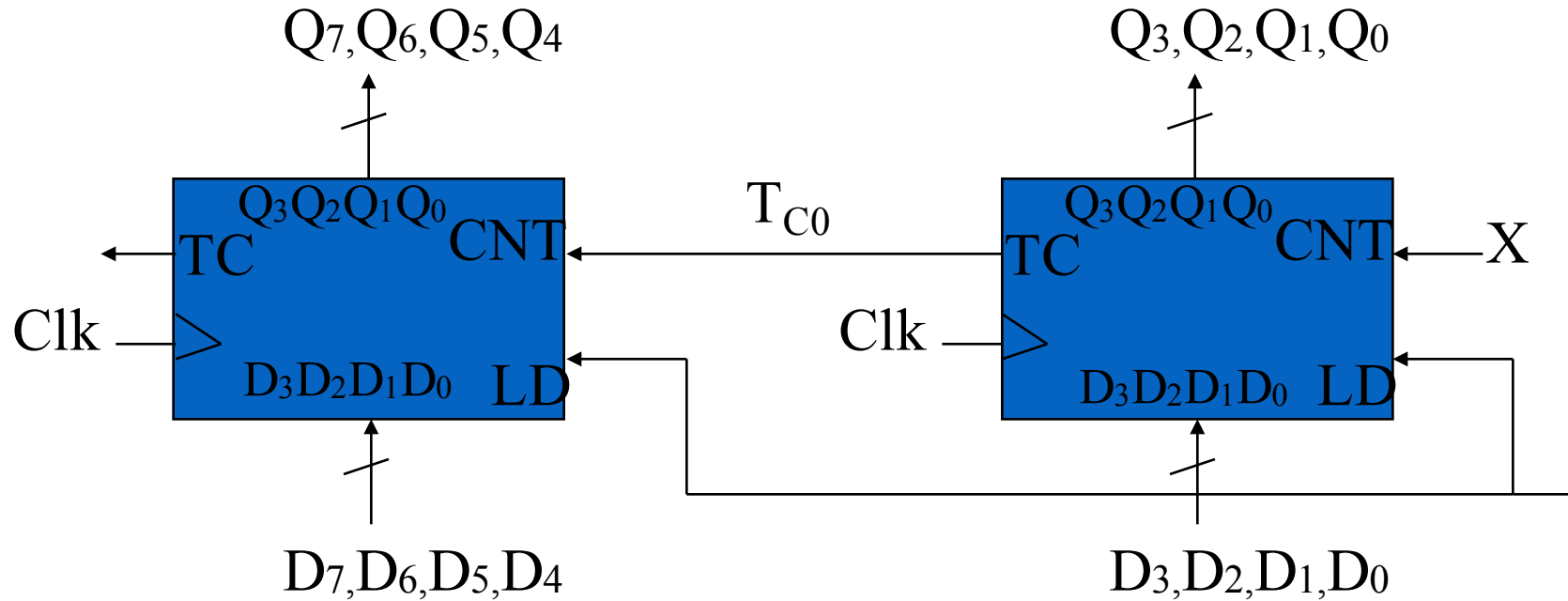
$$D_2 = Q_2'$$

$$D_0 = Q_3 + Q_2'$$

Id	$Q_3 Q_2 Q_1 Q_0$	LD	D_3	D_2	D_1	D_0
1	0001	0	-	-	-	-
2	0010	0	-	-	-	-
3	0011	1	0	1	0	1
5	0101	0	-	-	-	-
6	0110	0	-	-	-	-
7	0111	1	1	0	1	0
10	1010	0	-	-	-	-
11	1011	1	1	1	1	1
15	1111	1	0	0	0	1
0, 4, 8, 9, 12, 13, 14	0000 ...	-	-	-	-	-

Cascade Counter

- A Cascade Modulo 256 Counter constructed with Modulo 16 counter



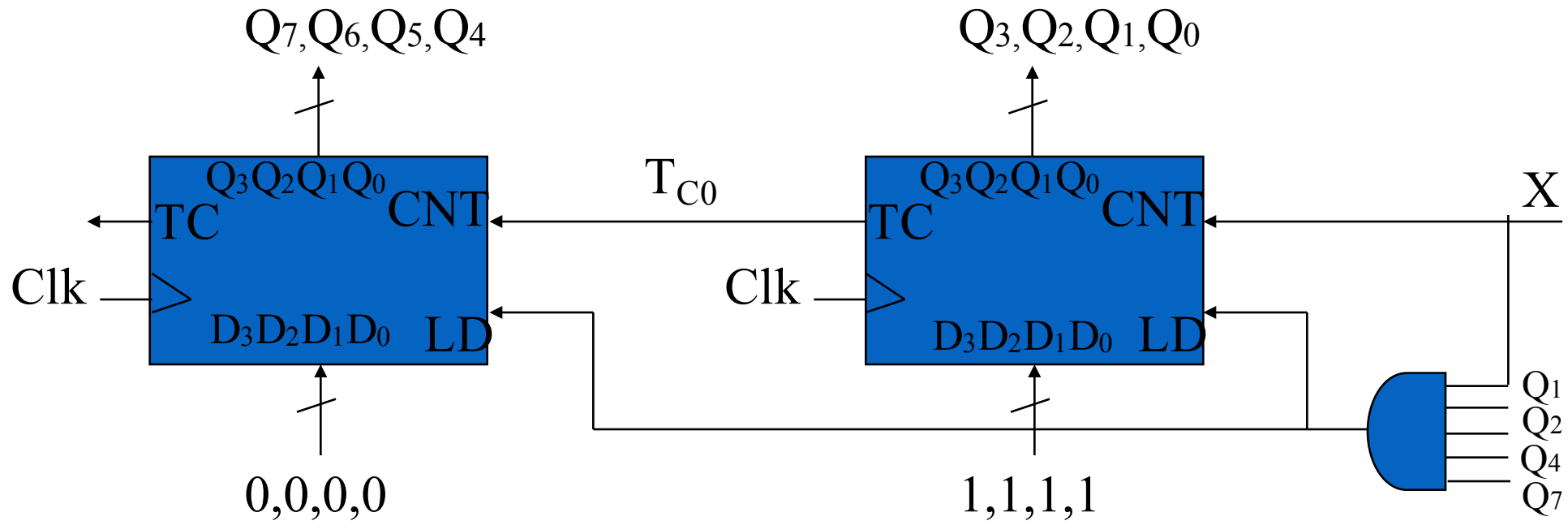
Cascade Counter

- The first modulo 16 counter
 - ✓ $T_{C0} = 1$ when $(Q_3, Q_2, Q_1, Q_0) = (1, 1, 1, 1)$ and $X=1$
- The second modulo 16 counter
 - ✓ $Q_{7:4}(t+1) = Q_{7:4}(t) + 1 \bmod 16$ when $T_{C0} = 1$
- The circuit functions as a modulo 256 counter.

Time		0	1	2	3	...	13	14	15	16	17	18	19
Q_{7-4}		0	0	0	0	...	0	0	0	1	1	1	1
T_{C0}		0	0	0	0	...	0	0	1	0	0	0	0
Q_{3-0}		0	1	2	3	...	13	14	15	0	1	2	3

Cascade Counter

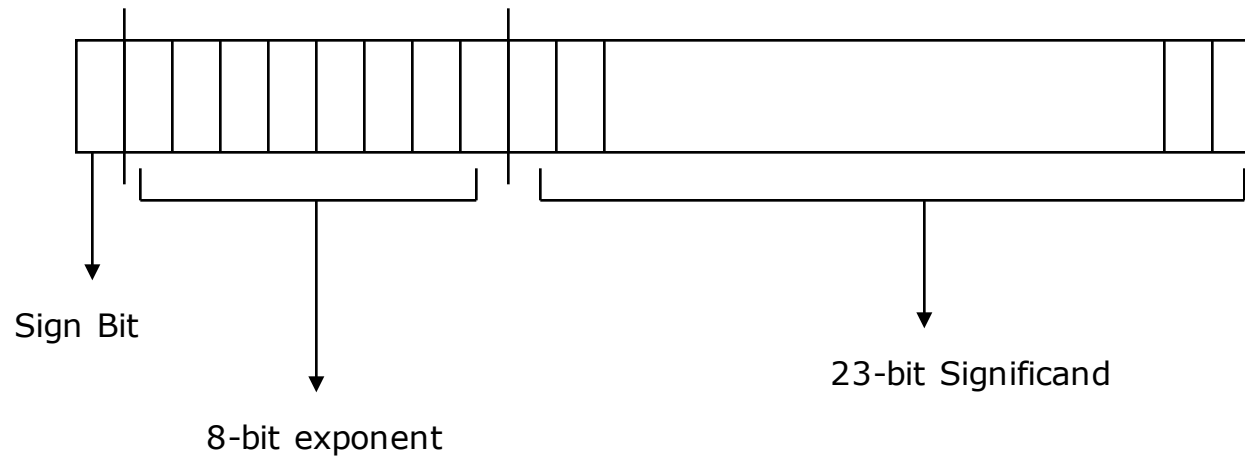
- Count from 15 to 150
 - ✓ Set $D_{7:0} = 15 = (0,0,0,0,1,1,1,1)$
 - ✓ When $X = 1$ and $Q_{7:0} = 150 = (1,0,0,1,0,1,1,0)$, set **LD** = 1 then $D_{7:0} \rightarrow Q_{7:0}$
 - ✓ Comb logic for LD only takes 1s



Floating Point Number: Standard

- ulp (**unit in the last place**)
 - Difference between two consecutive values of the significand.

3 Parts $\rightarrow x = \sim s b^e$: sign, significand, exponent



Standard: Normalization

- $\pm e_1 e_2 e_3 e_4 e_5 e_6 e_7 e_8 s_1 s_2 s_3 \dots s_{22} s_{23}$
- $1.s_1 s_2 s_3 \dots s_{22} s_{23}$ normalized number
- $0.s_1 s_2 s_3 \dots s_{22} s_{23}$ denormalized number
- Id $e_1 e_2 e_3 e_4 e_5 e_6 e_7 e_8$

Total numbers
= $2 \times 255 \times 2^{23} - 1$
(excluding Inf,
NaN)

0	0 0 0 0 0 0 0 0	$x = 0.s_1 s_2 s_3 \dots s_{22} s_{23} 2^{-126}$	→	Tiniest = 1×2^{-149}
1	0 0 0 0 0 0 0 1	$x = 1.s_1 s_2 s_3 \dots s_{22} s_{23} 2^{-126}$	→	Normalized minimum
2	0 0 0 0 0 0 1 0	$x = 1.s_1 s_2 s_3 \dots s_{22} s_{23} 2^{-125}$		$N_{\min} = 1.0 \times 2^{-126}$
.				
126	0 1 1 1 1 1 1 0	$x = 1.s_1 s_2 s_3 \dots s_{22} s_{23} 2^{-1}$		
127	0 1 1 1 1 1 1 1	$x = 1.s_1 s_2 s_3 \dots s_{22} s_{23} 2^0$		
128	1 0 0 0 0 0 0 0	$x = 1.s_1 s_2 s_3 \dots s_{22} s_{23} 2^1$		
.				
253	1 1 1 1 1 1 0 1	$x = 1.s_1 s_2 s_3 \dots s_{22} s_{23} 2^{126}$		Normalized maximum
254	1 1 1 1 1 1 1 0	$x = 1.s_1 s_2 s_3 \dots s_{22} s_{23} 2^{127}$	→	$N_{\max} = (2 - 2^{-23}) 2^{127}$
255	1 1 1 1 1 1 1 1	$x = \text{Inf if } (s_1 \dots s_{23}) = 0, \text{ else NaN}$		

NaN → Not a Number

Conventional Rounding Error

Rounding Error

$$1.s_1s_2s_3\dots s_{22}s_{23}00 \rightarrow 1.s_1s_2s_3\dots s_{22}s_{23} = 0$$

$$1.s_1s_2s_3\dots s_{22}s_{23}01 \rightarrow 1.s_1s_2s_3\dots s_{22}s_{23} = -0.25$$

$$1.s_1s_2s_3\dots s_{22}s_{23}10 \rightarrow 1.s_1s_2s_3\dots s_{22}s_{23} = +0.5$$

$$1.s_1s_2s_3\dots s_{22}s_{23}11 \rightarrow 1.s_1s_2s_3\dots s_{22}s_{23} = +0.25$$

Average Absolute Error
 $= |-0.25| + 0.5 + 0.25 / 4 = 0.25$

- Average Absolute Error $err_{abs} = |-0.25| + 0.5 + 0.25 / 4 = 0.25 \cdot 2^{-23}$

- Relative Error

- The real number x represented as floating number (32 bits) $B[x]_2$ is between $1.0_10_20_3\dots 0_{22}0_{23}$ and $1.1_11_21_3\dots 1_{22}1_{23} \approx 1.0_10_20_3\dots 0_{22}0_{23} 2^1$

- The expected relative error $= \frac{|x - B[x]_2|}{|x|} = \frac{|err_{abs}|}{|x|}$

$$\frac{1}{2} \times 0.25 \times 2^{-23} \leq err_{rel} \leq 1 \times 0.25 \times 2^{-23}$$

$$1.49 \times 10^{-8} \leq err_{rel} \leq 2.98 \times 10^{-8}$$

Rounding: round to even

- Round up conditions: 1.bbb...bbbG**R****X****X****X****X**
 - Round = 1, Sticky = 1 $\rightarrow$ > 0.5
 - Guard = 1, Round = 1, Sticky = 0 $\rightarrow$ Round to even
- examples

Value	Fraction	G R S	Incr?	Rounded
128	1.0000 0 000	000	N	1.000
15	1.101 0 000	100	N	1.101
17	1.000 1 000	010	N	1.000
19	1.001 1 000	110	Y	1.010
138	1.000 1 010	011	Y	1.001
63	1.111 1 100	111	Y	10.000

Round up= R(G+S)

Guard bit: LSB of result

Round bit: 1st bit removed

Sticky bit: OR of remaining bits

Rounding: Round to even

- 1.bbb...bbbG^RS

- Round up = R(G+S)
- G^RS Rouding Error

000	0	0
001	0	-0.25
010	0	-0.5
011	+1	+0.25
100	0	0
101	0	-0.25
110	+1	+0.5
111	+1	+0.25

Average Error = 0

Absolute Average Error = 0.25

Guard bit: LSB of result

Round bit: 1st bit removed

Sticky bit: OR of remaining bits

Conventional Rounding error

Rounding	Error
1.bbbG00 → 1.101	= 0
1.bbbG01 → 1.101	= -0.25
1.bbbG10 → 1.110	= +0.5
1.bbbG11 → 1.110	= +0.25

Average Error = 0.125

Absolute Average Error = 0.25

Floating Point Number

- Addition: $\sim s_1 x b^{e_1} + (\sim s_2 x b^{e_2}) = \sim s x b^e$
 $= \sim s_1 x b^{e_1} + \sim s_2 / b^{e_1 - e_2} x b^{e_1}$
 $= (\sim s_1 + \sim s_2 / b^{e_1 - e_2}) x b^{e_1}$
- Multiplication: $(\sim s_1 x b^{e_1}) x (\sim s_2 x b^{e_2}) = \sim (s_1 x s_2) b^{e_1 + e_2}$
- Practice

- $1.101_2 \times 2^{-110} \times 1.011_2 \times 2^{-80}$ underflow
- $1.110_2 \times 2^{110} \times 1.101_2 \times 2^{80}$ overflow
- $1.101_2 \times 2^{10} - 1.01101011010_2 \times 2^{-10}$

1

1.101

10

-0.000000000000000000000000101101011010

23

1.100111111111111111111110100**10100110**

G R S

2^{10}

2^{10}

2^{10}

- $1.101_2 \times 2^{110} / (1.011_2 \times 2^{-70})$ **overflow**

G=0, R=1, S=1

Result =

$$1.100111111111111111110101_2 \times 2^{10}$$