Digital Electronics and Microprocessor Systems (ELEC211)

Dave McIntosh and Valerio Selis

dmc@liv.ac.uk

v.selis@liv.ac.uk

Digital 7: Simple registers and counters

UNIVERSITY OF LIVERPOOL

Outline

- Registers
- Parallel adder (with accumulator)
- Counters

Previous material

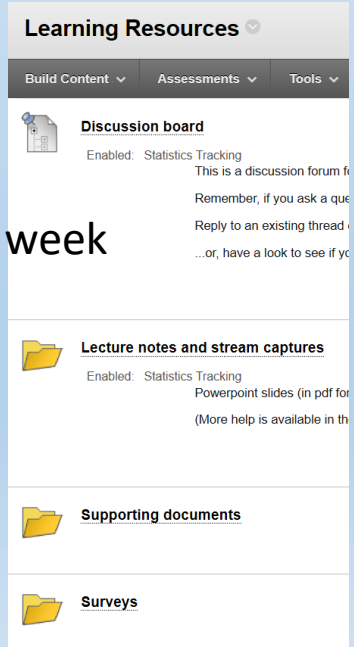
Flip-flops ✓

Karnaugh maps ✓

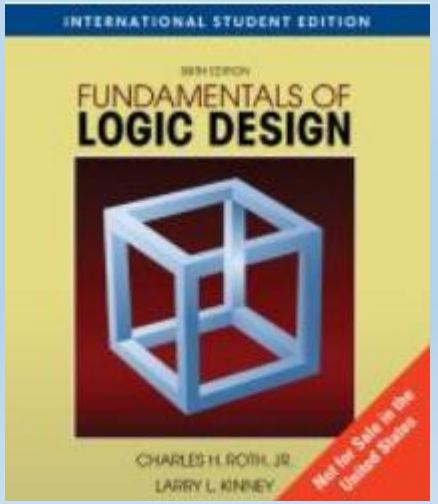
Use VITAL!:

- Stream lectures
- Handouts
- Notes and Q&A each week
- Discussion Board
- Exam resources

www.liv.ac.uk/vital



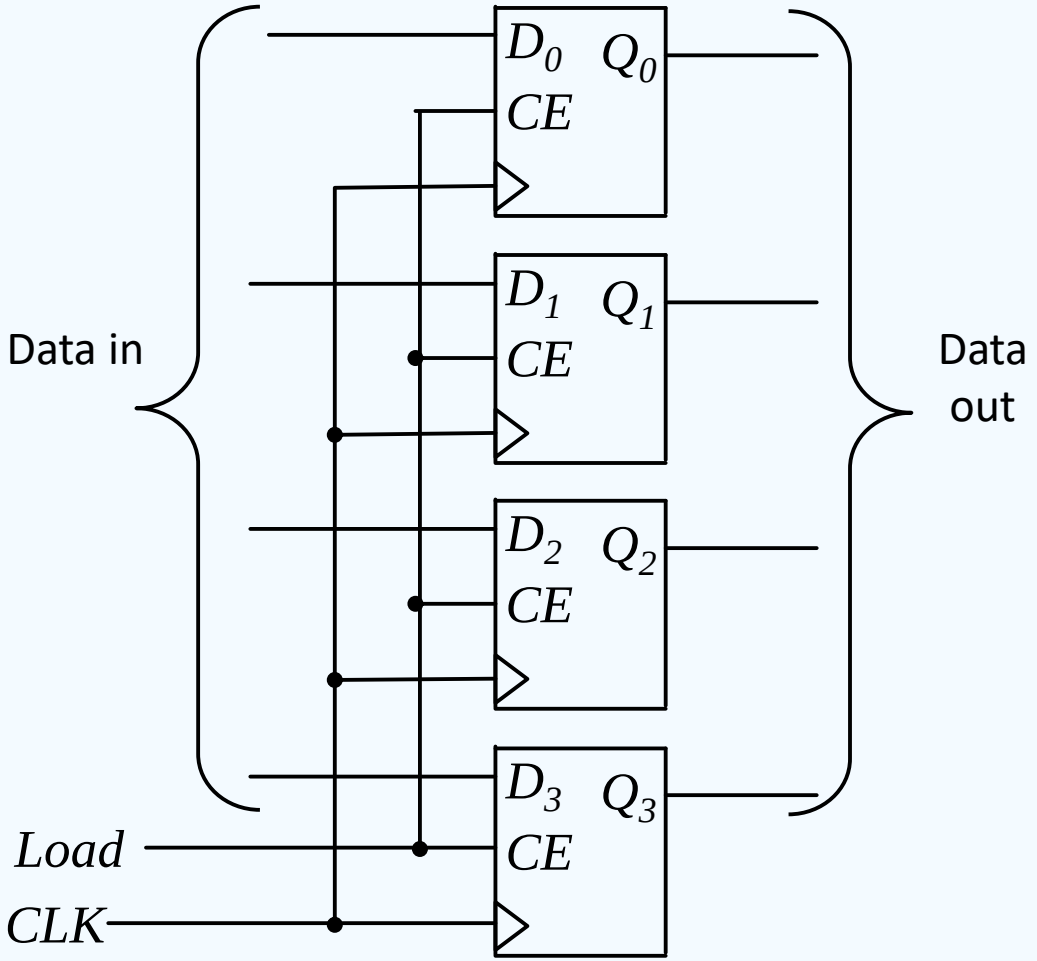
Course textbook – please borrow and use it!



UNIVERSITY OF LIVERPOOL

Registers

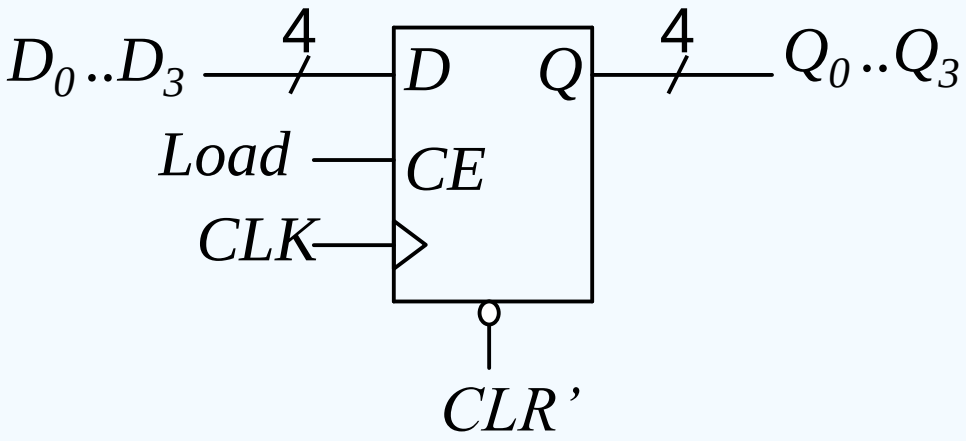
- Several D-Type flip-flops may be grouped with a common clock to form a register.
- Each flip-flop stores 1 bit
- This register stores 4 bits



Registers

The flip-flops in the register can have a common asynchronous clear input, *CLR'*.

Bubble at the input indicates that a logic 0 is required to clear the flip-flops.

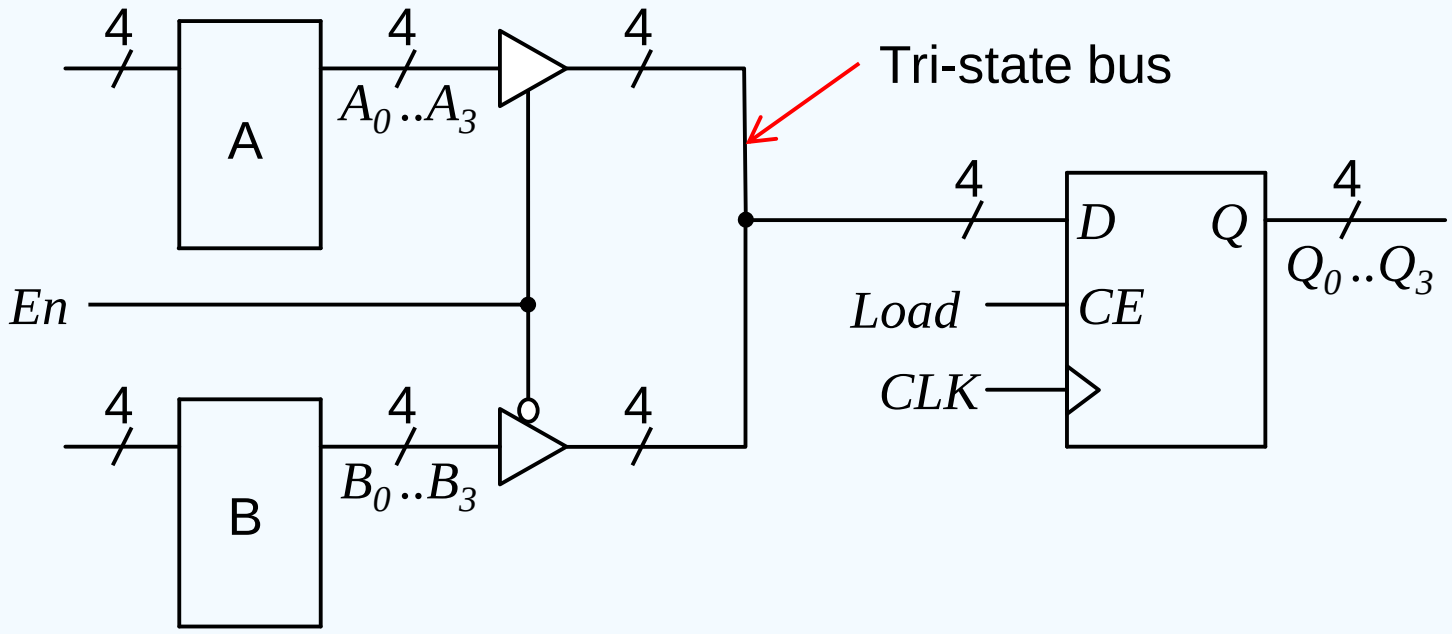


When *Load* is 1 the clock is enabled (*CE*) and the data applied to the *D* inputs will be loaded into the flip-flops. When *Load* is 0 the register holds its data (memory).

Register symbol is the same as the D-type flip-flop except for the bus input and output.

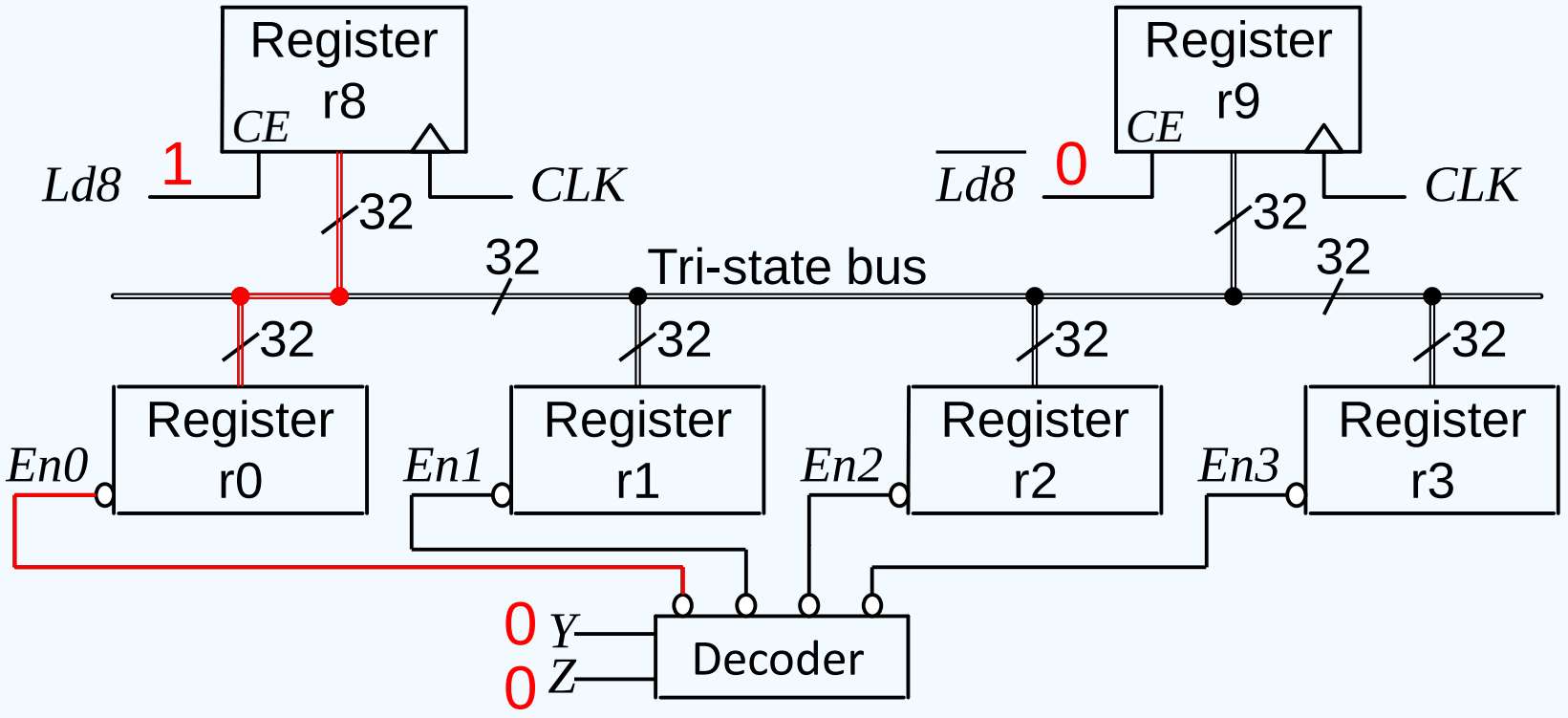
Data Transfer between Registers

If $En = 1$ and $Load = 1$ the output of register A is enabled onto the tri-state bus and will be stored in Q register on the rising-edge of the clock.



If $En = 0$ and $Load = 1$ the output of register B is stored in Q register.

Data Transfer using a Three State Bus



32 bits of data are transferred in parallel from register r0, r1, r2 or r3 to register r8 or r9. If $YZ = 00$ and $Ld8 = 1$ data in r0 is stored in r8. ARM instruction: MOV r8, r0

Remember the 1-bit adder?

X	Y	C_{in}	C_{out}	Sum
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

- Adds two bits and a carry (C_{in})
- Generates a Sum and another carry (C_{out})

- Can connect 4 together to form a **4-bit** (parallel) **adder**
- C_{out} from first, becomes C_{in} of second, etc…

Adding binary numbers – an aside

Consider the binary operation $0101_2 + 0111_2$ (equivalent to 5 + 7 in decimal form). To formally add the two binary numbers, we have to carry in the same way as we would using column addition of decimal numbers:

$$\begin{array}{r} 0\ 1\ 0\ 1 \\ +\ 0\ 1\ 1\ 1 \\ \hline \end{array}$$

1 1 1

1 1 0 0

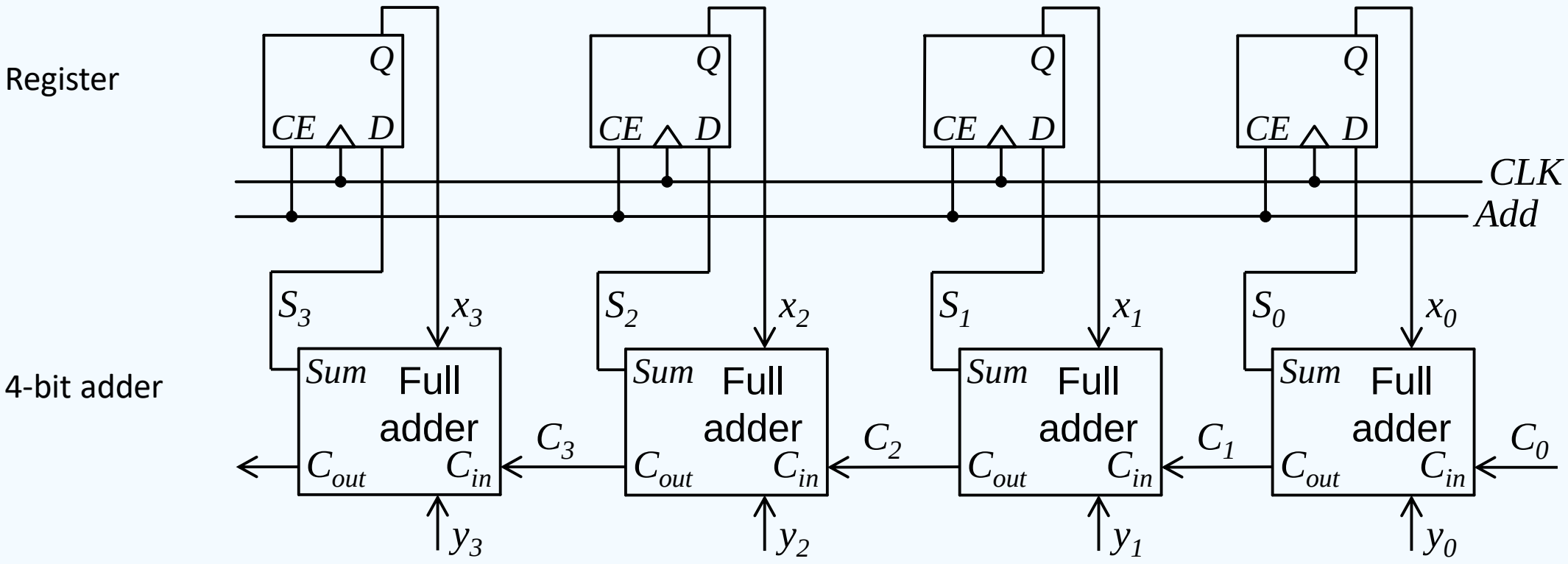
The red 1s are “carried” 1s. The answer is 1100_2 , or $12_{(10)}$ in decimal form.

Adding binary numbers – an aside

$$
\begin{array}{r}
0\ 1\ 0\ 1 \\
+\ 0\ 1\ 1\ 1 \\
\hline
\color{red}{1\ \ 1\ \ 1\ \ \ } \\
1\ 1\ 0\ 0
\end{array}
$$

- With a 4-bit adder, we are trying to automate the process above.
- The truth table in the previous slide lists the eight possible combinations that could occur in a given column (except the first or right-hand column, which in the example here, has no possibility of a carry-in.)
- X and Y each represent a 4-bit number i.e. one of the top two rows in the calculation above.

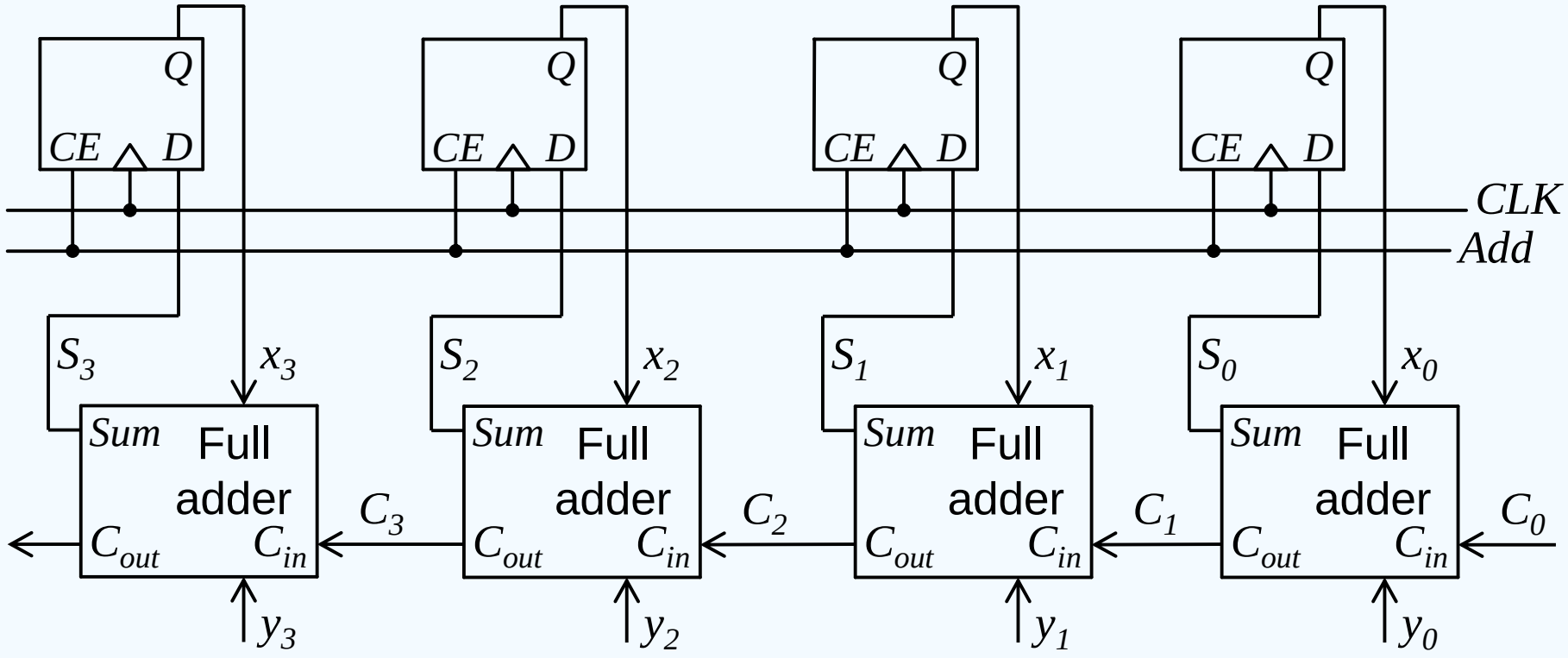
Parallel Adder with Accumulator



In computer circuits, it's often desirable to add a number to the already-stored content of a register of flip-flops (which is called an accumulator)

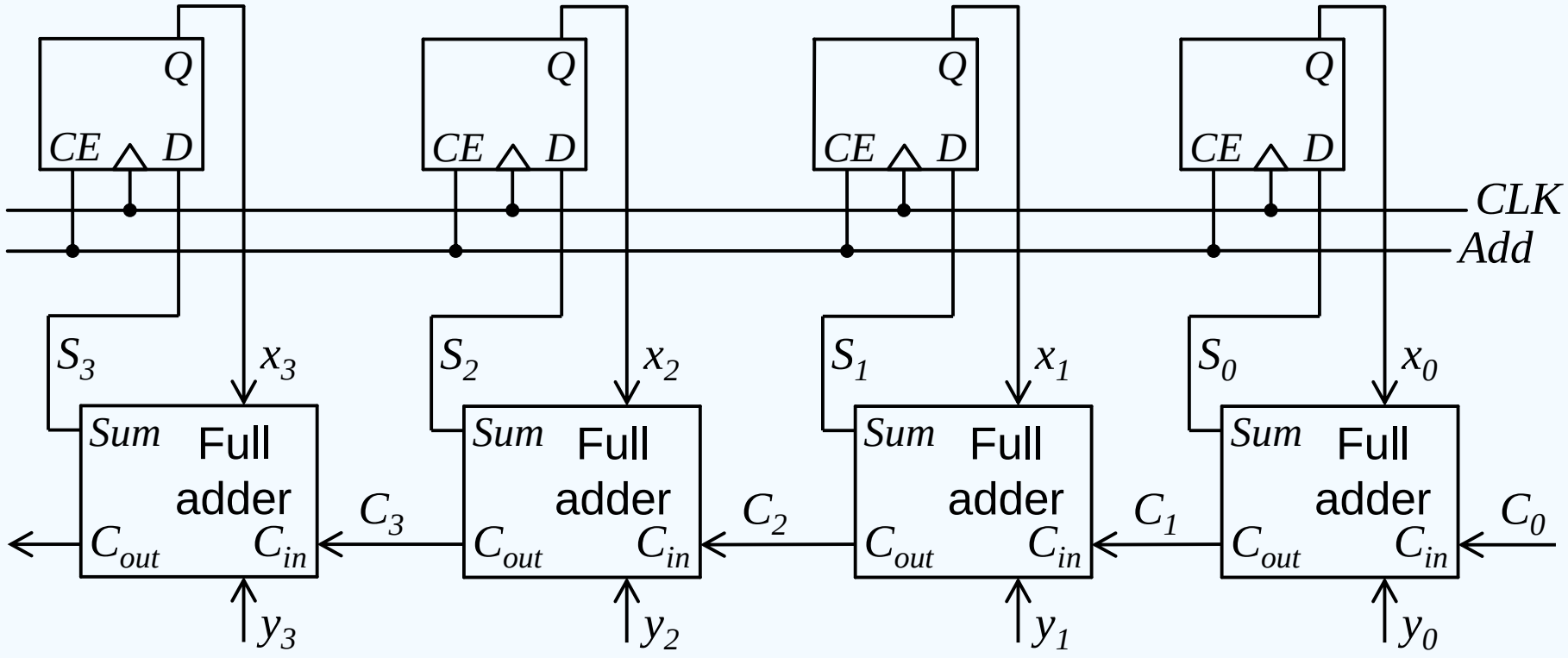
e.g. the ARM instruction: ADD rx, rx, ry

Parallel Adder with Accumulator



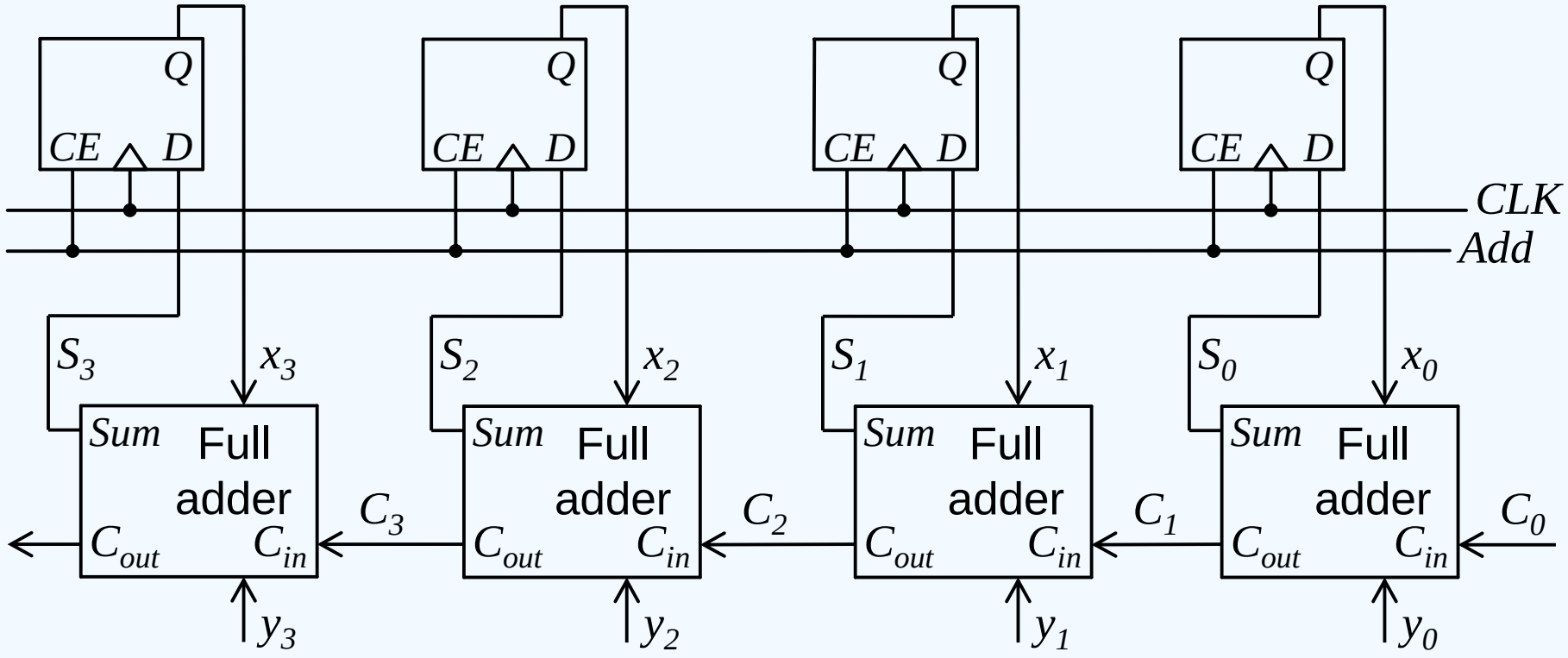
In the adder circuit, the output of the accumulator, X= $x_3 \ldots x_0$ is added to the input number, Y= $y_3 \ldots y_0$

Parallel Adder with Accumulator



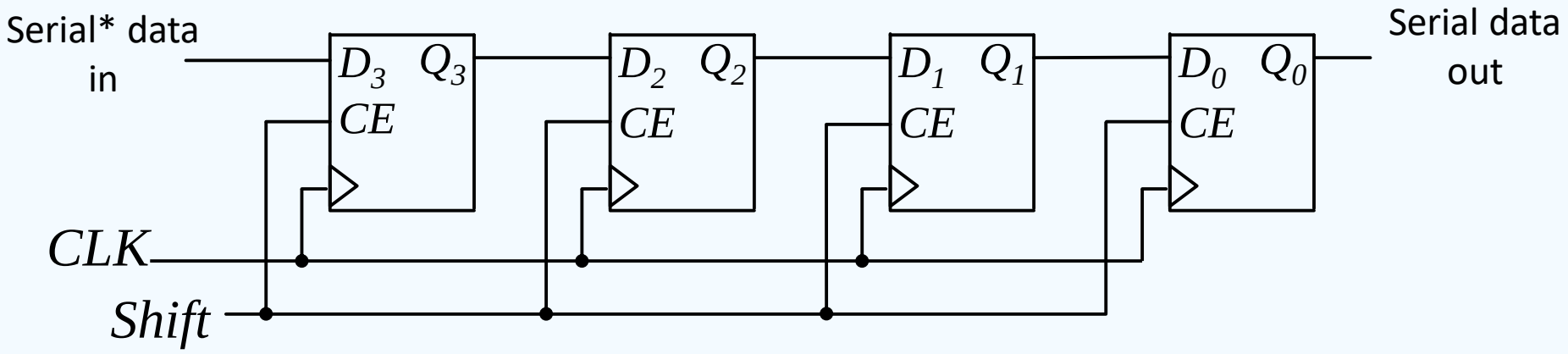
After the carry has propagated through the adders, the sum, $S_3 \ldots S_0$, will be stabilised.

Parallel Adder with Accumulator



If Add = 1, the number in X= $x_3 \ldots x_0$ in the accumulator is replaced with S = $S_3 \ldots S_0$ at the next active clock edge.

Shift Registers

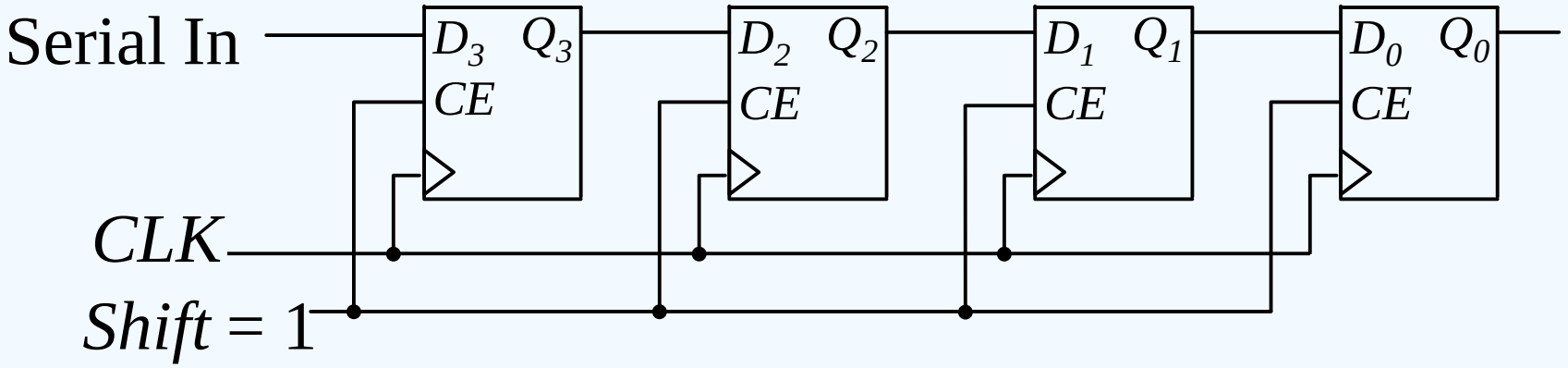


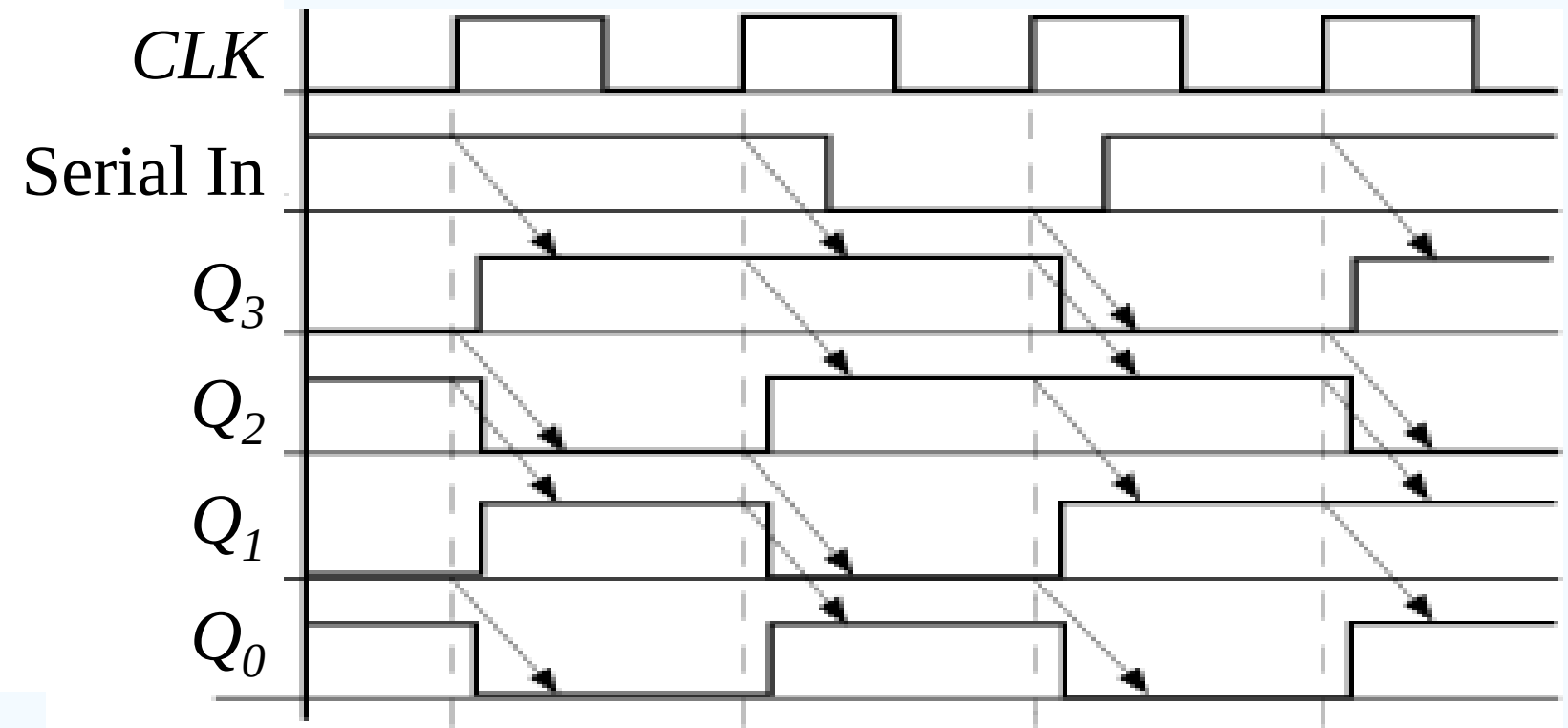
A shift register is a register in which binary data can be stored, and shifted to the right or left.

When $Shift = 1$, the clock is enabled and shifting to the right occurs on the rising clock edge.

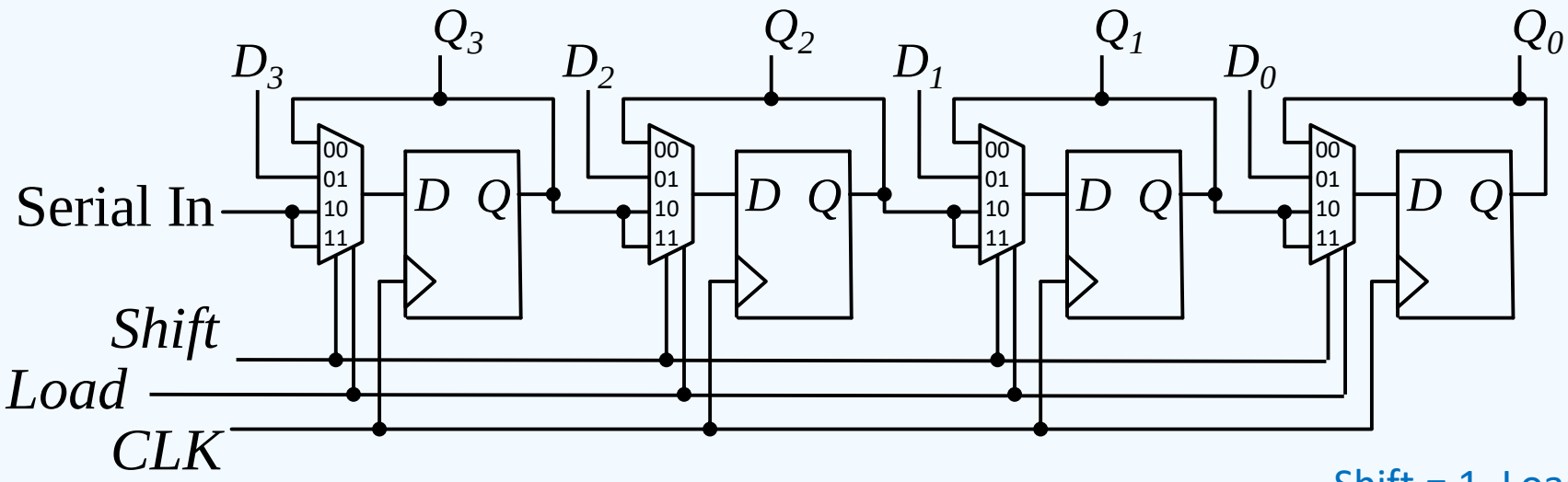
When $Shift = 0$ no shifting occurs.

* Serial in: Data is 'shifted' onto the first FF **one bit, then the next, etc.**; FFs not loaded in parallel
Serial out: data can only be read out of the last FF (no other FF connections to IC terminals)

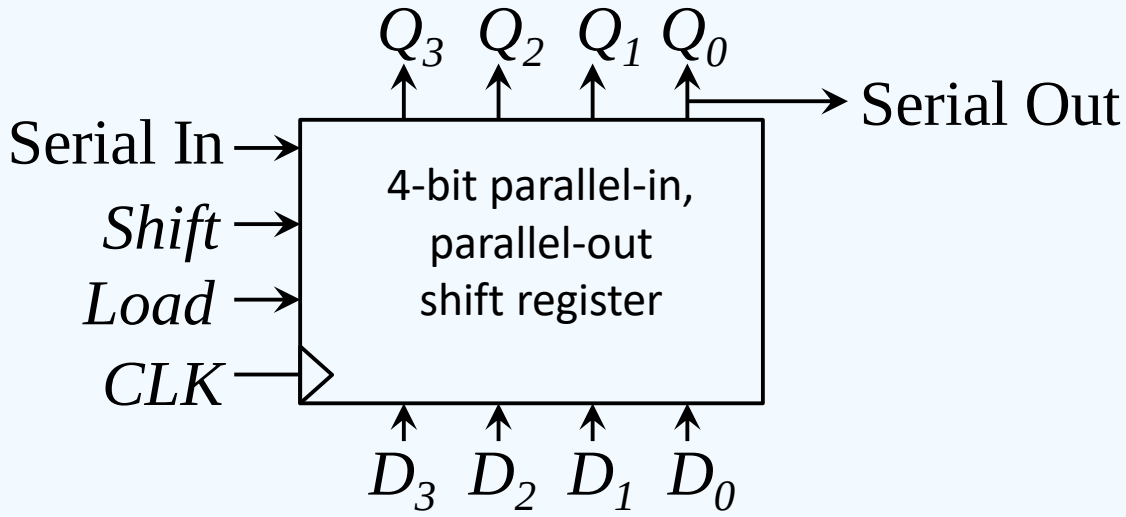

Serial In
D_3 Q_3
CE
D_2 Q_2
CE
D_1 Q_1
CE
D_0 Q_0
CE
CLK
Shift = 1


CLK
Serial In
Q_3
Q_2
Q_1
Q_0

Parallel-in, Parallel-out Shift Register



Block diagram:

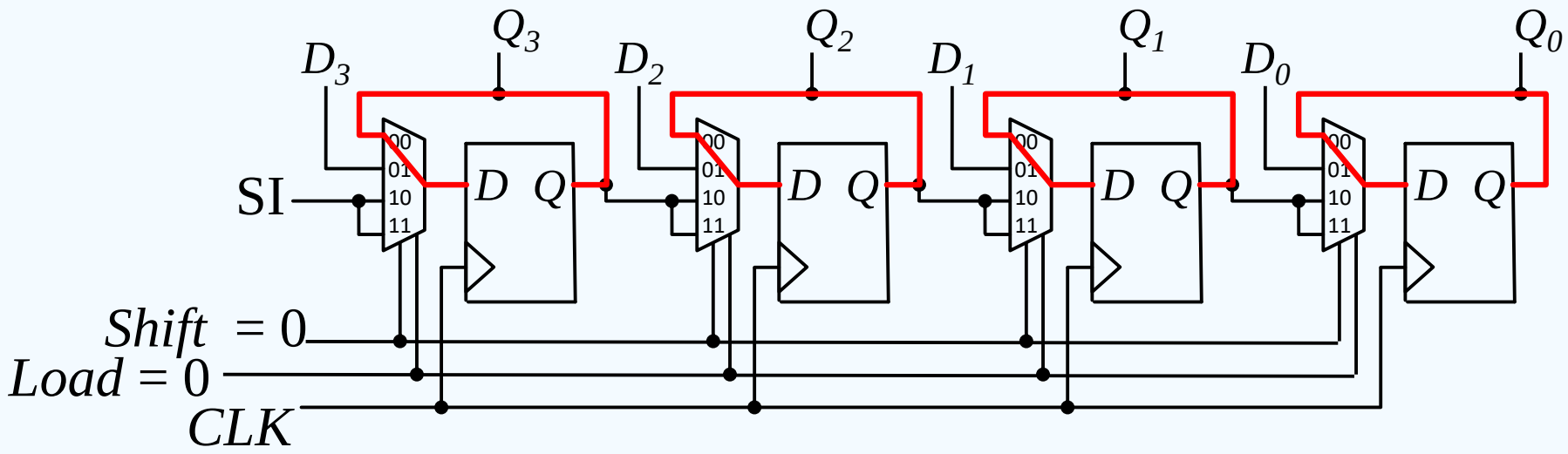


Shift = 1, Load = 1 or 0:
Serial input is shifted into first FF

Shift = 0, Load = 1:
The 4 data inputs are loaded in parallel

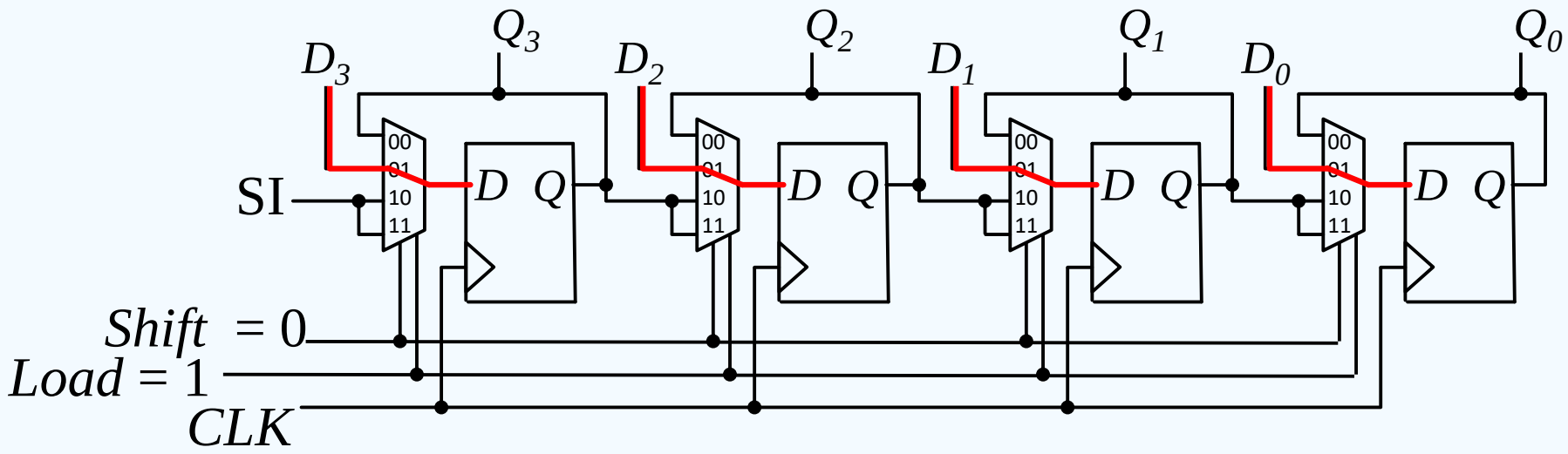
Shift = 0, Load = 0:
No change of state with the clock edge

Parallel-in, Parallel-out Shift Register



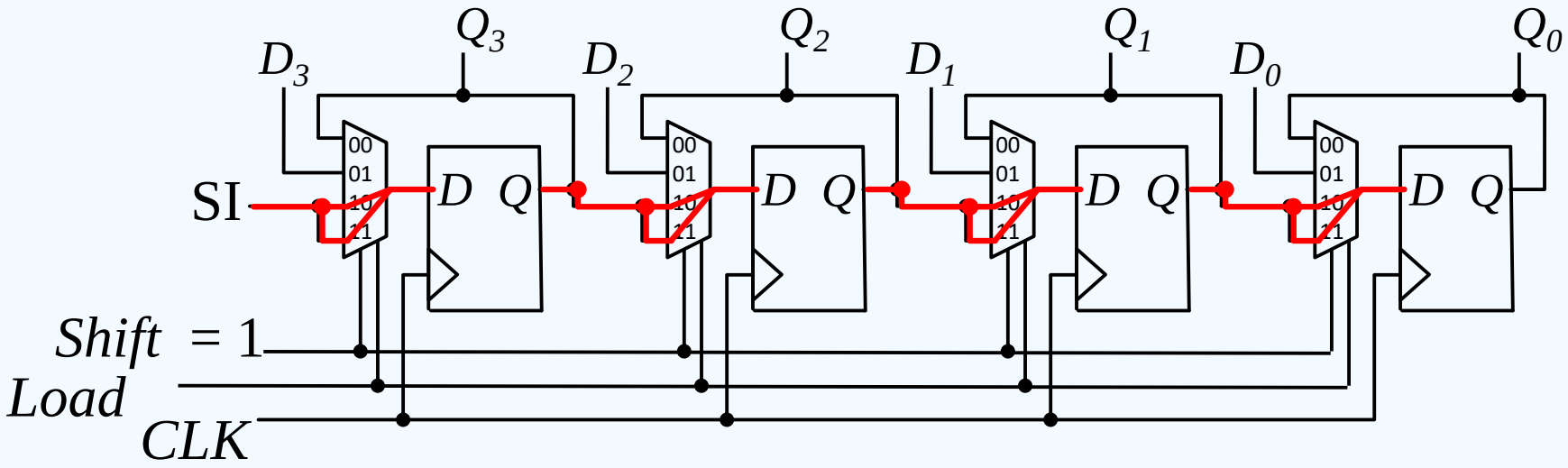
Inputs		Next state				Action
Shift	*Load*	Q^+_3	Q^+_2	Q^+_1	Q^+_0	
0	**0**	Q_3	Q_2	Q_1	Q_0	**no change**
0	1	D_3	D_2	D_1	D_0	load
1	X	SI	Q_3	Q_2	Q_1	right shift

Parallel-in, Parallel-out Shift Register



Inputs		Next state				Action
Shift	*Load*	Q^+_3	Q^+_2	Q^+_1	Q^+_0	
0	0	Q_3	Q_2	Q_1	Q_0	no change
0	**1**	D_3	D_2	D_1	D_0	**load**
1	X	SI	Q_3	Q_2	Q_1	right shift

Parallel-in, Parallel-out Shift Register



Inputs		Next state				Action
Shift	*Load*	Q^+_3	Q^+_2	Q^+_1	Q^+_0	
0	0	Q_3	Q_2	Q_1	Q_0	no change
0	1	D_3	D_2	D_1	D_0	load
1	**X**	**SI**	$\boldsymbol{Q_3}$	$\boldsymbol{Q_2}$	$\boldsymbol{Q_1}$	**right shift**

Synchronous Binary Counters using D-type Flip-Flops

Outputs of flip-flops determine the state of the counter.

Binary counter restarts at zero after it reaches all one state.

A 3-bit counter can be designed as follows…

Present State			Next state		
C	B	A	C^+	B^+	A^+
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

Binary Counter using D-type FF

Present State			Next state		
C	B	A	C^+	B^+	A^+
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	0	0	0	0

C^+

BA \ C	0	1
00	0	1
01	0	1
11	1	0
10	0	1

B^+

BA \ C	0	1
00	0	0
01	1	1
11	0	0
10	1	1

A^+

BA \ C	0	1
00	1	1
01	0	0
11	0	0
10	1	1

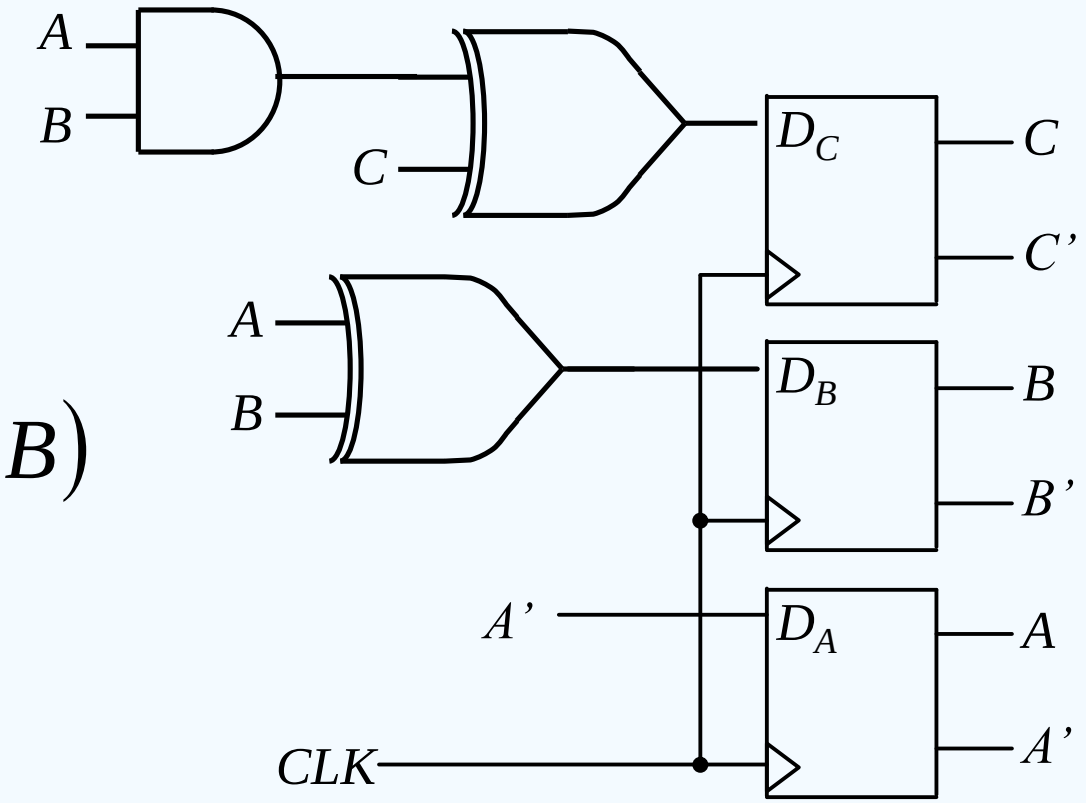
$$D_C = C^+ = \overline{C}.B.A + C.\overline{B} + C.\overline{A}$$
$$= C \oplus (A.B)$$
$$D_B = B^+ = \overline{B}.A + B.\overline{A} = A \oplus B$$
$$D_A = A^+ = \overline{A}$$

Binary Counter using D-type FF

$$D_C = C \oplus (A.B)$$

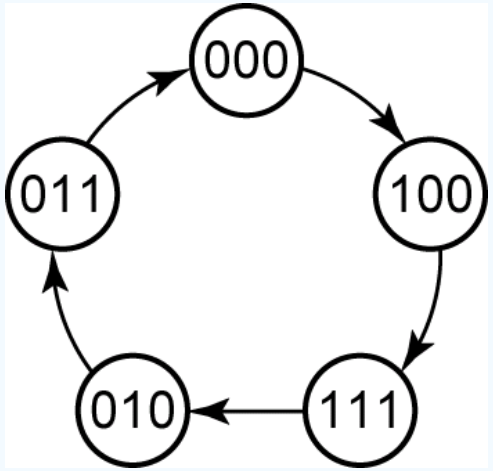
$$D_B = A \oplus B$$

$$D_A = \overline{A}$$



Counters for other sequences using D-type Flip-Flops

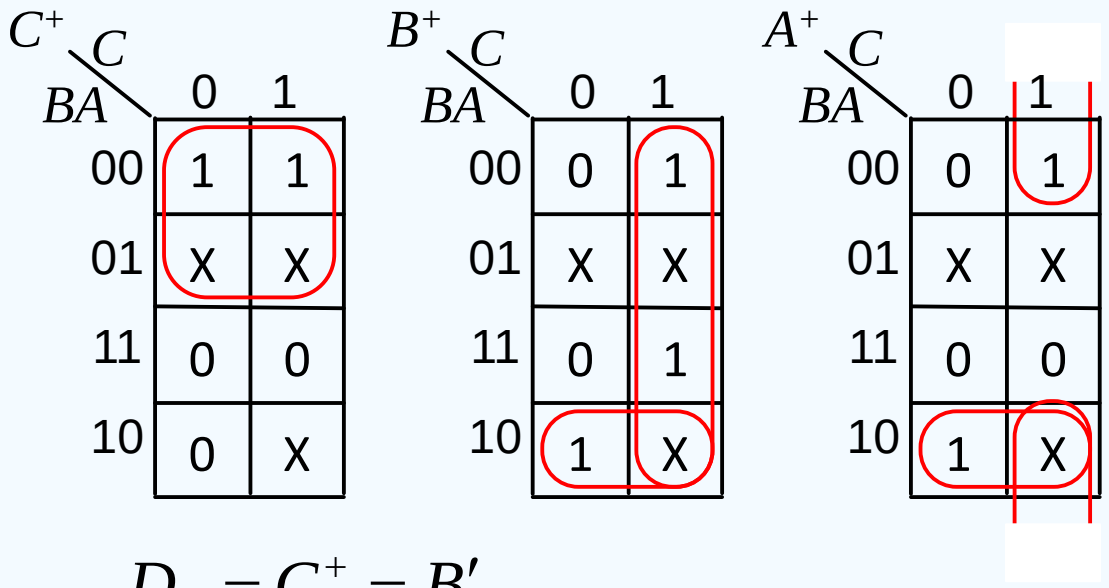
In some applications the sequence of states of a counter is not in strict binary order.



Present State			Next state		
C	B	A	C^+	B^+	A^+
0	0	0	1	0	0
0	0	1	-	-	-
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	-	-	-
1	1	0	-	-	-
1	1	1	0	1	0

Counters for other sequences using D-type flip-flops

Present State			Next state		
C	B	A	C^+	B^+	A^+
0	0	0	1	0	0
0	0	1	-	-	-
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	-	-	-
1	1	0	-	-	-
1	1	1	0	1	0



$$D_C = C^+ = B'$$

$$D_B = B^+ = C + BA'$$

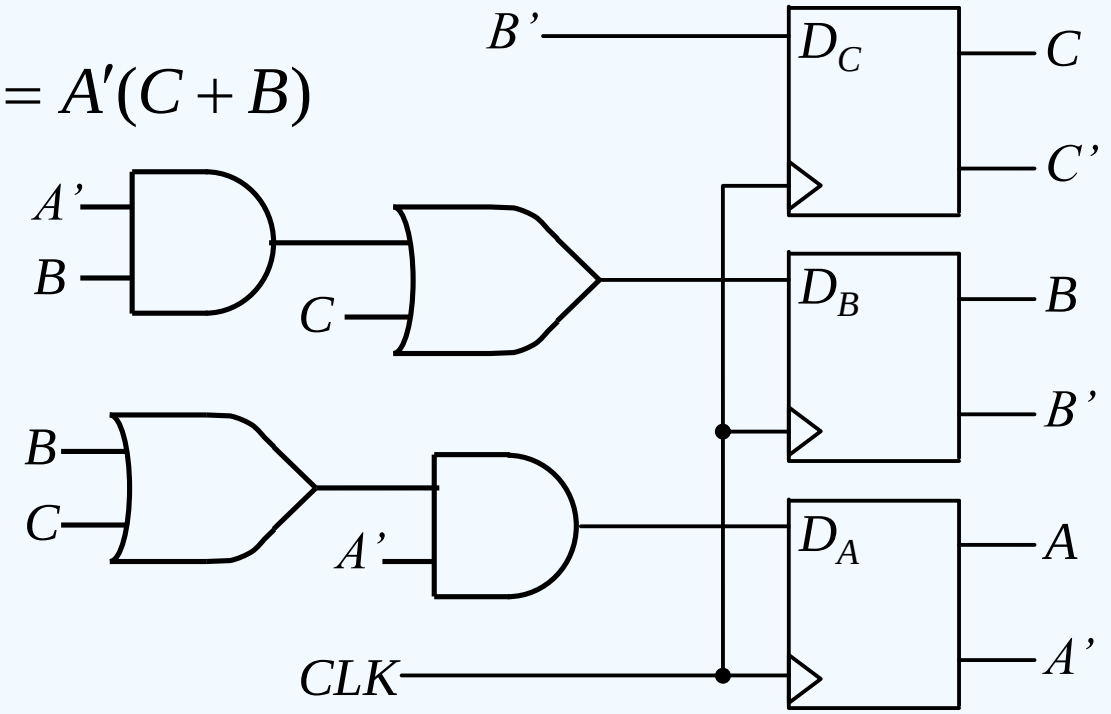
$$D_A = A^+ = CA' + BA' = A'(C + B)$$

Counters for other sequences using D-type flip-flops

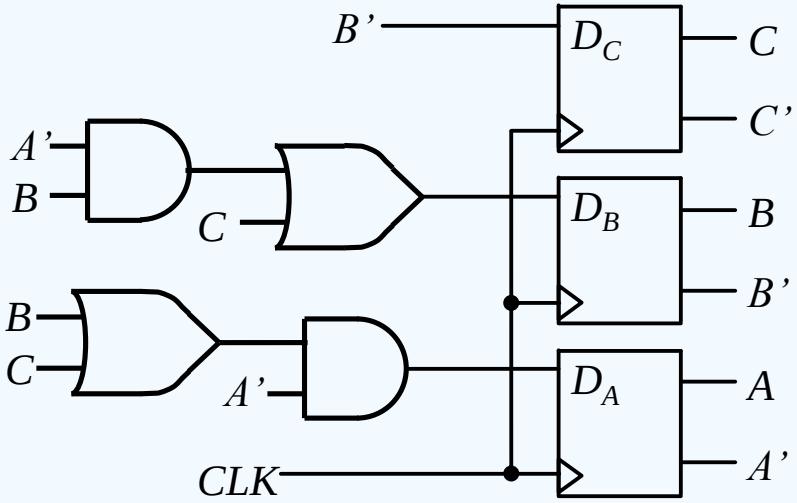
$$D_C = B'$$

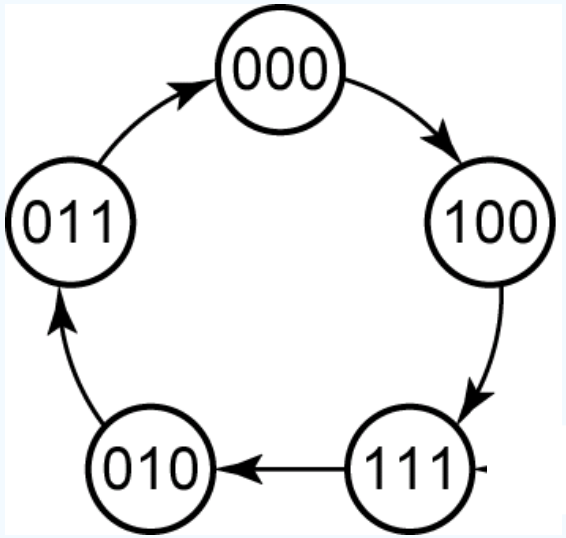
$$D_B = C + BA'$$

$$D_A = CA' + BA' = A'(C + B)$$



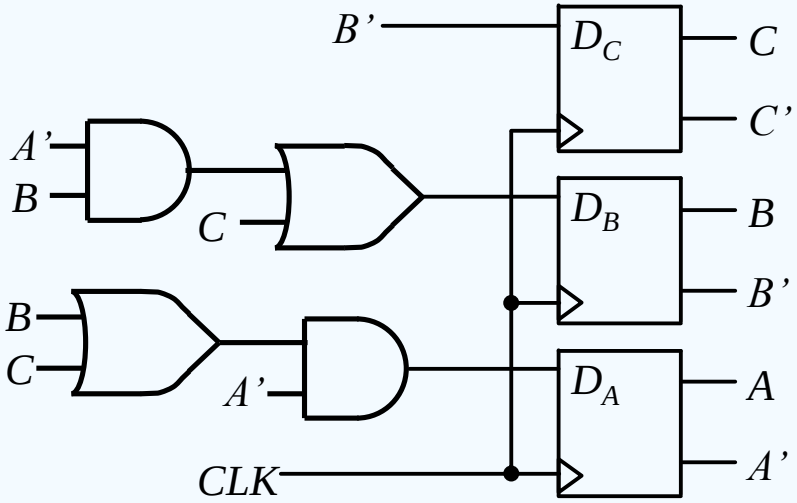
Counters: unused states

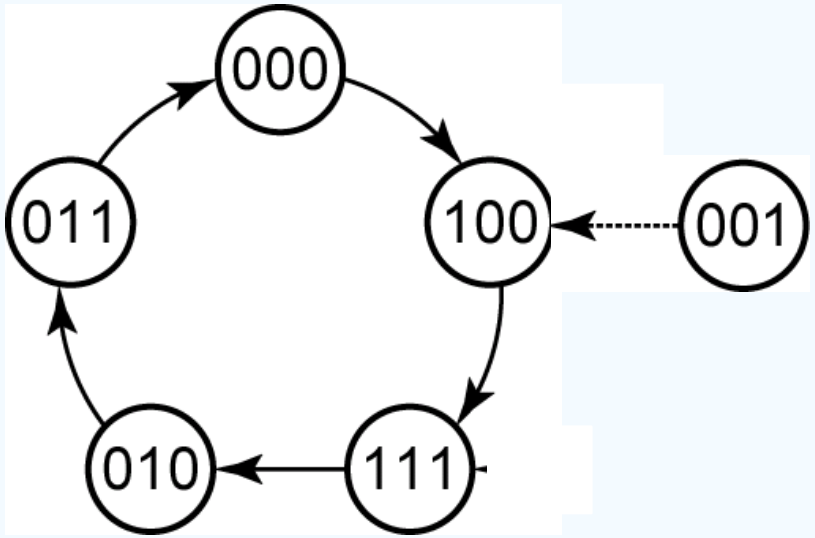




After power-on, the initial state of the flip-flops is unpredictable. If the initial state is CBA =101 then according to the circuit the next state value will be 110 which is not one of the valid states. Fortunately after the next clock pulse the output will become 011 which is a valid state.

Counters: unused states

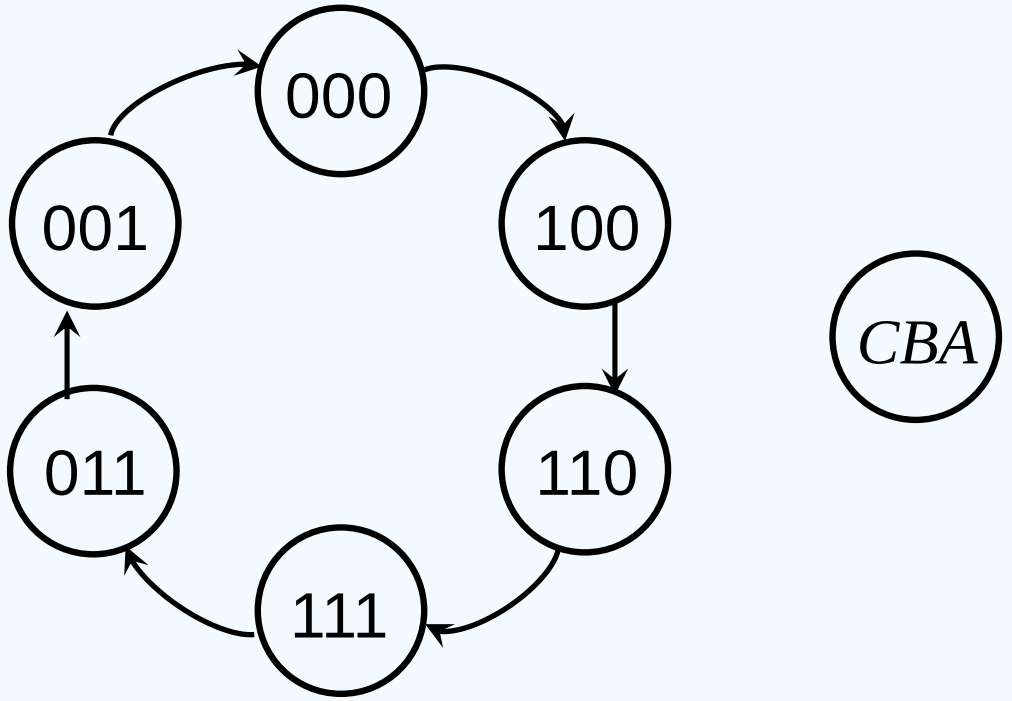




All don't care states should be checked to make sure that they eventually lead into the main counting sequence.

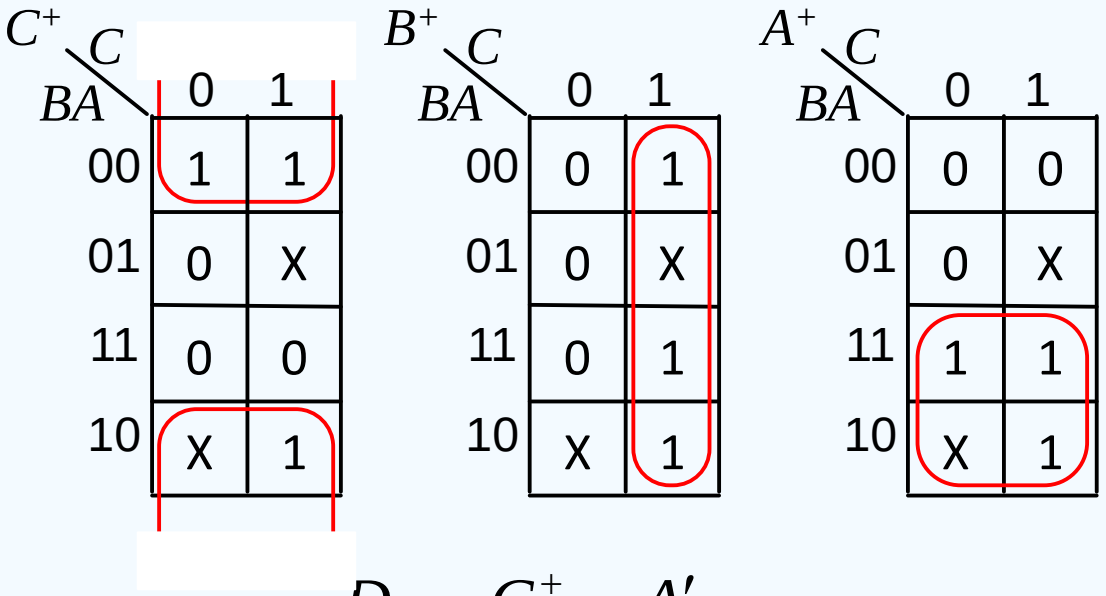
Question

Design a synchronous counter using D-type flip-flops for the following sequence (known as Gray code).



Answer

Present State			Next state		
C	B	A	C^+	B^+	A^+
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	--	--	--
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	--	--	--
1	1	0	1	1	1
1	1	1	0	1	1

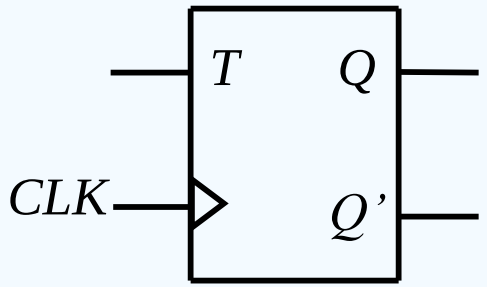


$$D_C = C^+ = A'$$

$$D_B = B^+ = C$$

$$D_A = A^+ = B$$

T or Toggle Flip-Flop – reminder:



T	Q	Q^+
0	0	0
0	1	1
1	0	1
1	1	0

T	Action at next clock pulse
0	No change, $Q^+ = Q$
1	Toggle, $Q^+ = Q'$

Synchronous Binary Counters using Toggle flip-flops

Outputs of flip-flops determine the state of the counter.

Counter rolls back to zero when it reaches all one state.

A 3-bit counter can be designed.

Present State			Next state			Flip-flop Inputs		
C	B	A	C^+	B^+	A^+	T_C	T_B	T_A
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

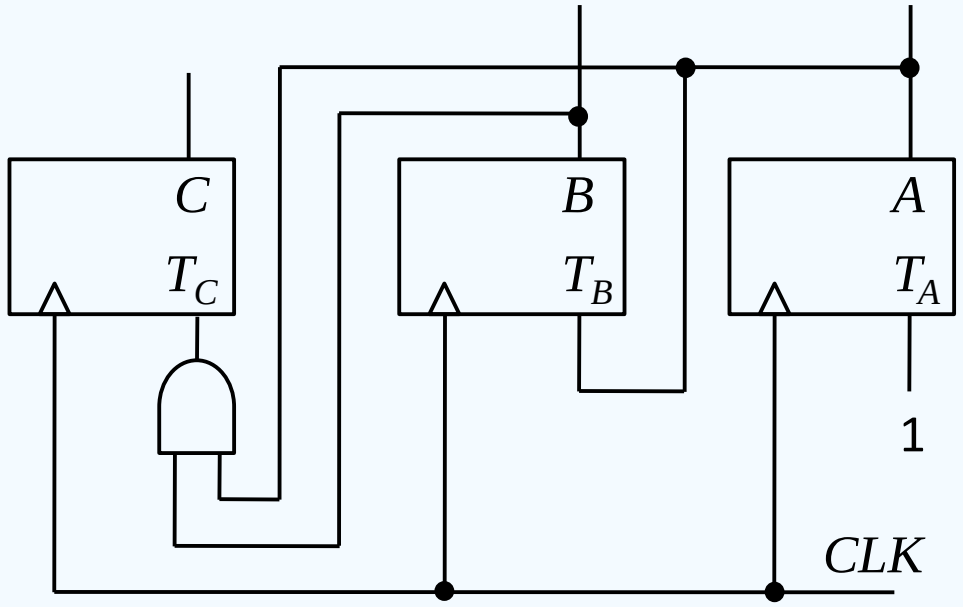
Binary Counter using T flip-flops

Present State			Next state			Flip-flop Inputs		
C	B	A	C^+	B^+	A^+	T_C	T_B	T_A
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

A changes state every time a clock pulse is received therefore $T_A = 1$.

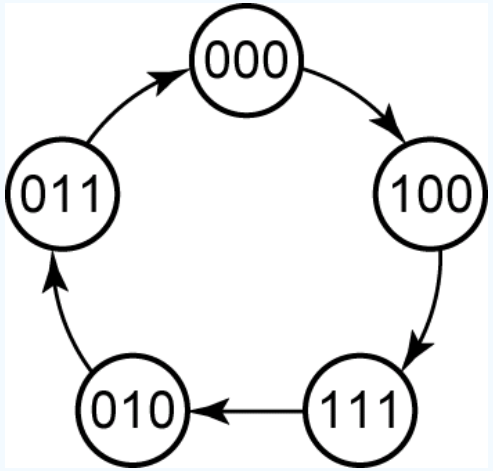
B changes only if $A = 1$, so $T_B = A$.

C changes only if $A = 1$ & $B = 1$, so $T_C = AB$.



Counters using T flip-flops

In some applications the sequence of states of a counter is not in strict binary order.

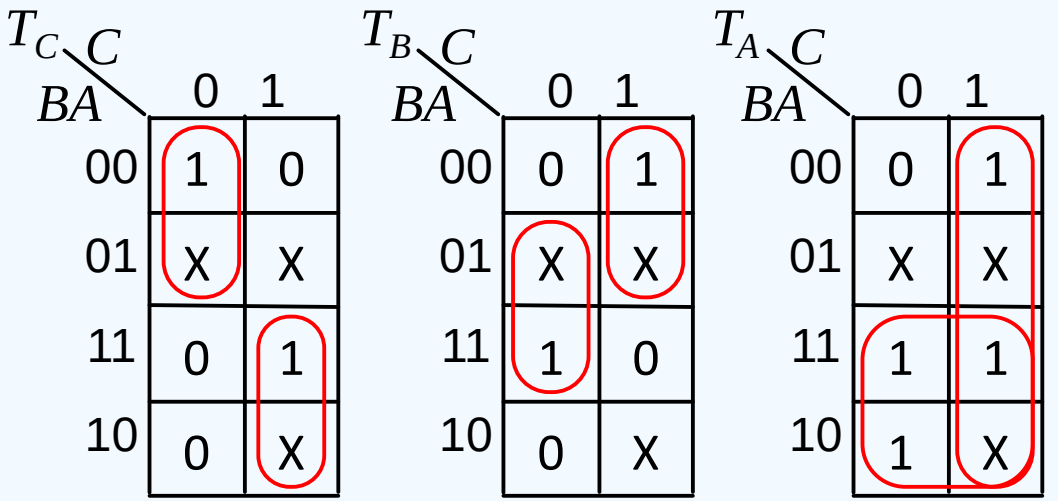


State graph

Present State			Next state			Flip-flop Inputs		
C	B	A	C^+	B^+	A^+	T_C	T_B	T_A
0	0	0	1	0	0	1	0	0
0	0	1	--	--	--	X	X	X
0	1	0	0	1	1	0	0	1
0	1	1	0	0	0	0	1	1
1	0	0	1	1	1	0	1	1
1	0	1	--	--	--	X	X	X
1	1	0	--	--	--	X	X	X
1	1	1	0	1	0	1	0	1

Counters using T flip-flops

Present State			Next state			Flip-flop Inputs		
C	B	A	C^+	B^+	A^+	T_C	T_B	T_A
0	0	0	1	0	0	1	0	0
0	0	1	--	--	--	X	X	X
0	1	0	0	1	1	0	0	1
0	1	1	0	0	0	0	1	1
1	0	0	1	1	1	0	1	1
1	0	1	--	--	--	X	X	X
1	1	0	--	--	--	X	X	X
1	1	1	0	1	0	1	0	1



$$T_C = C'B' + CB$$

$$T_B = C'A + CB'$$

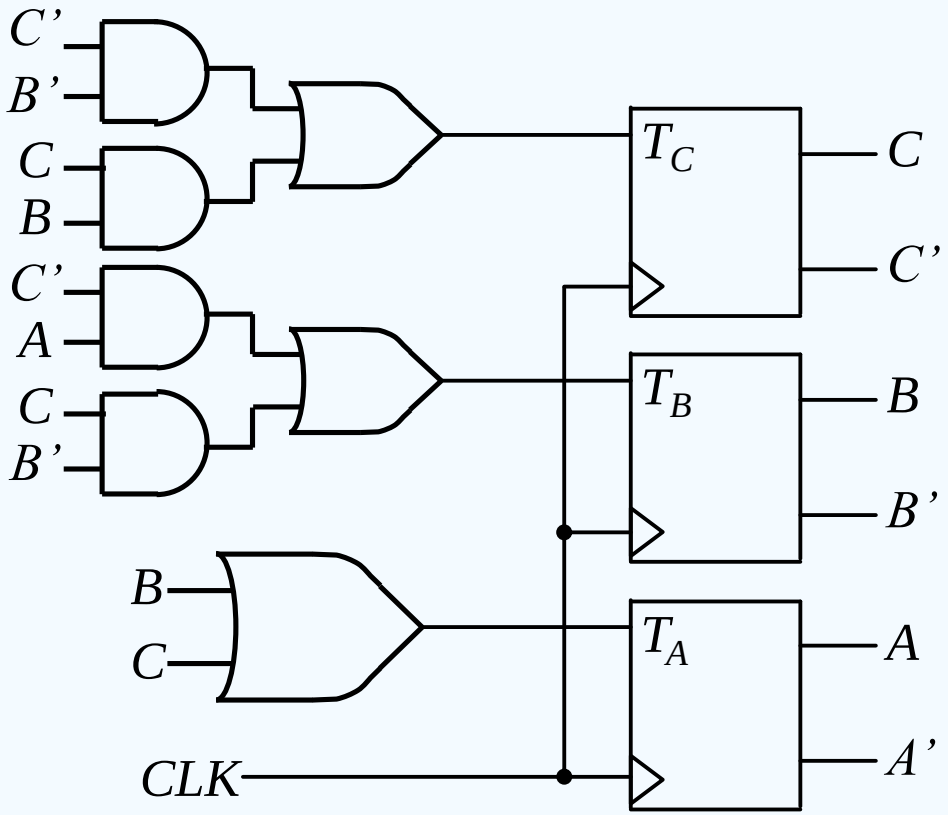
$$T_A = C + B$$

Counters using T flip-flops

$$T_C = C'B' + CB$$

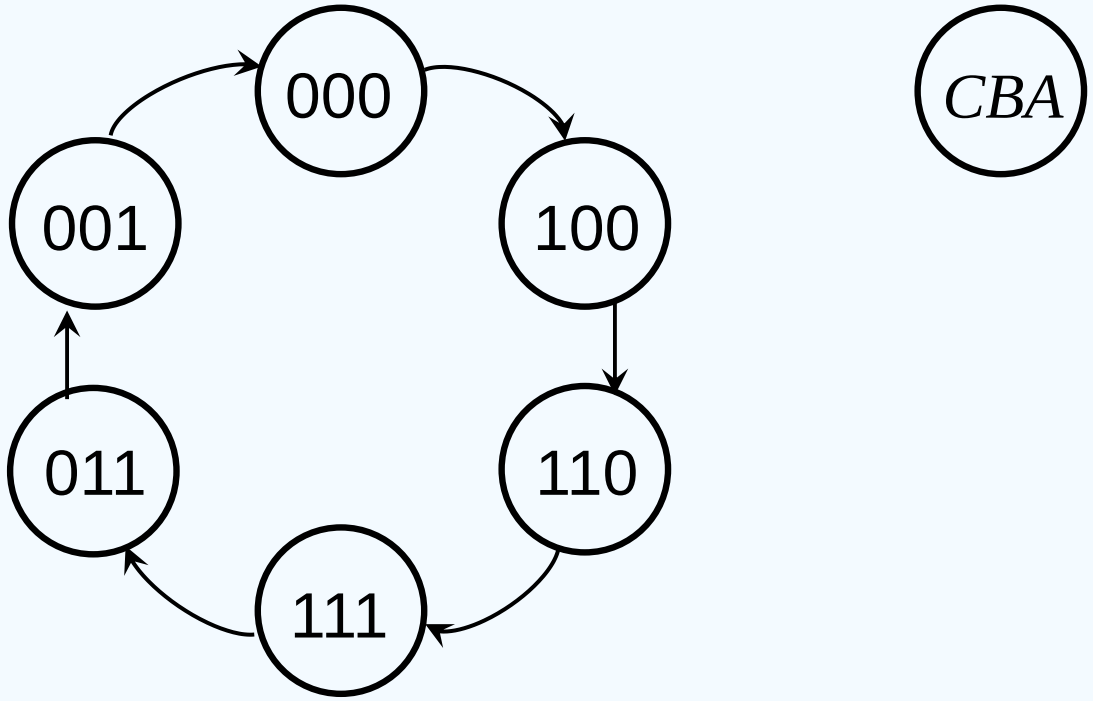
$$T_B = C'A + CB'$$

$$T_A = C + B$$



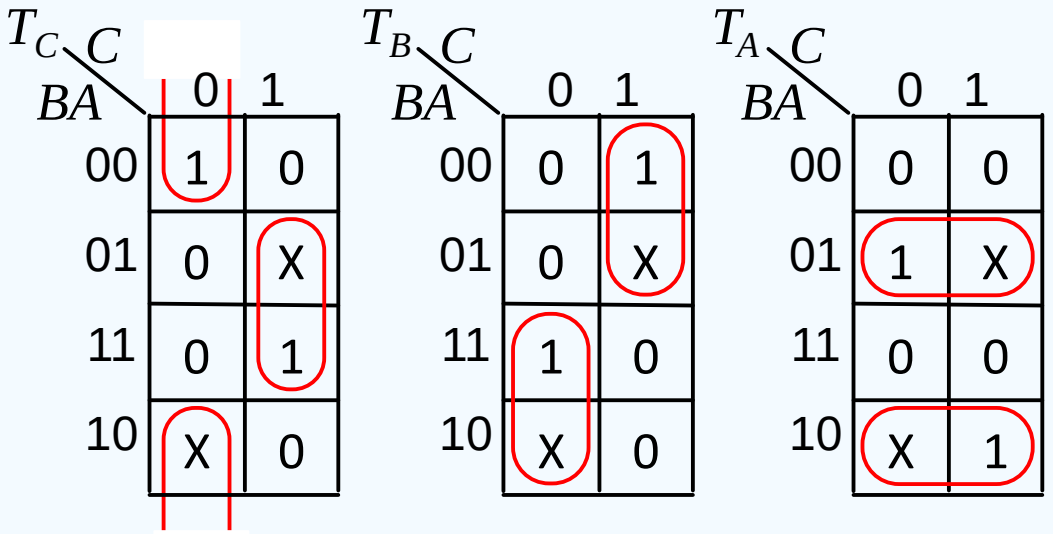
Question

Design a synchronous counter for the following sequence (Gray code) using T flip-flops.



Answer

Present State			Next state			Flip-flop Inputs		
C	B	A	C^+	B^+	A^+	T_C	T_B	T_A
0	0	0	1	0	0	1	0	0
0	0	1	0	0	0	0	0	1
0	1	0	--	--	--	X	X	X
0	1	1	0	0	1	0	1	0
1	0	0	1	1	0	0	1	0
1	0	1	--	--	--	X	X	X
1	1	0	1	1	1	0	0	1
1	1	1	0	1	1	1	0	0



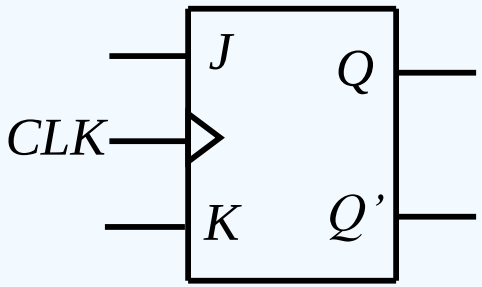
$$T_C = C'A' + CA = \overline{C \oplus A}$$

$$T_B = CB' + C'B = C \oplus B$$

$$T_A = B'A + BA' = B \oplus A$$

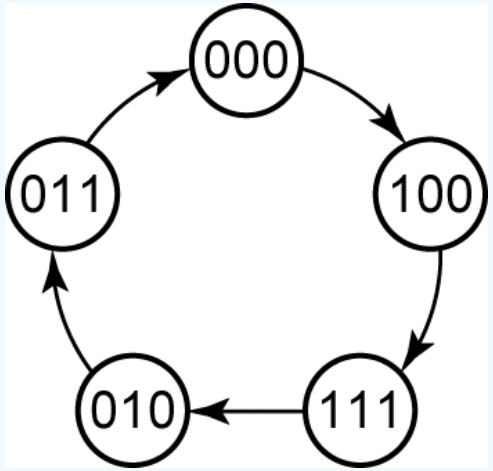
J-K Flip-Flop - reminder

J	K	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



J	K	Action at next clock pulse
0	0	No change, $Q^+ = Q$
0	1	Reset, $Q^+ = 0$
1	0	Set, $Q^+ = 1$
1	1	Toggle, $Q^+ = Q'$

Counter Design using J-K Flip-Flops



Present State			Next state		
C	B	A	C^+	B^+	A^+
0	0	0	1	0	0
0	0	1	--	--	--
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	--	--	--
1	1	0	--	--	--
1	1	1	0	1	0

J-K flip-flop excitation table

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

UNIVERSITY OF LIVERPOOL

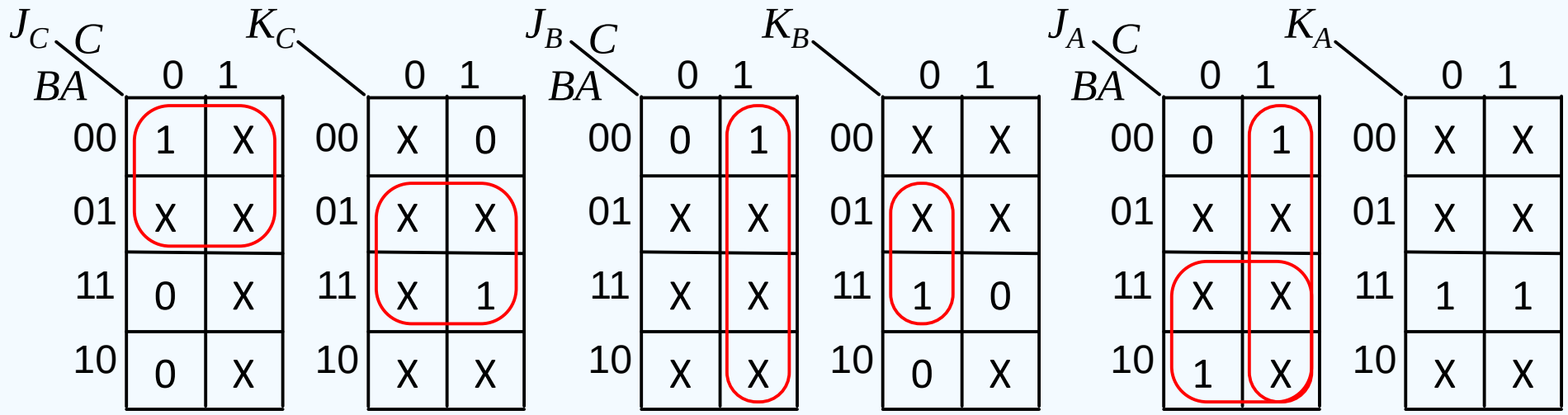
Counter Design using J-K Flip-Flops

Present State			Next state			Flip-flop inputs					
C	B	A	C^+	B^+	A^+	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	1	0	0	1	X	0	X	0	X
0	0	1	--	--	--	X	X	X	X	X	X
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	0	0	0	0	X	X	1	X	1
1	0	0	1	1	1	X	0	1	X	1	X
1	0	1	--	--	--	X	X	X	X	X	X
1	1	0	--	--	--	X	X	X	X	X	X
1	1	1	0	1	0	X	1	X	0	X	1

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

J-K flip-flop excitation table

Counter Design using J-K Flip-Flops



$$J_C = B'$$
$$K_C = A$$

$$J_B = C$$
$$K_B = C'A$$

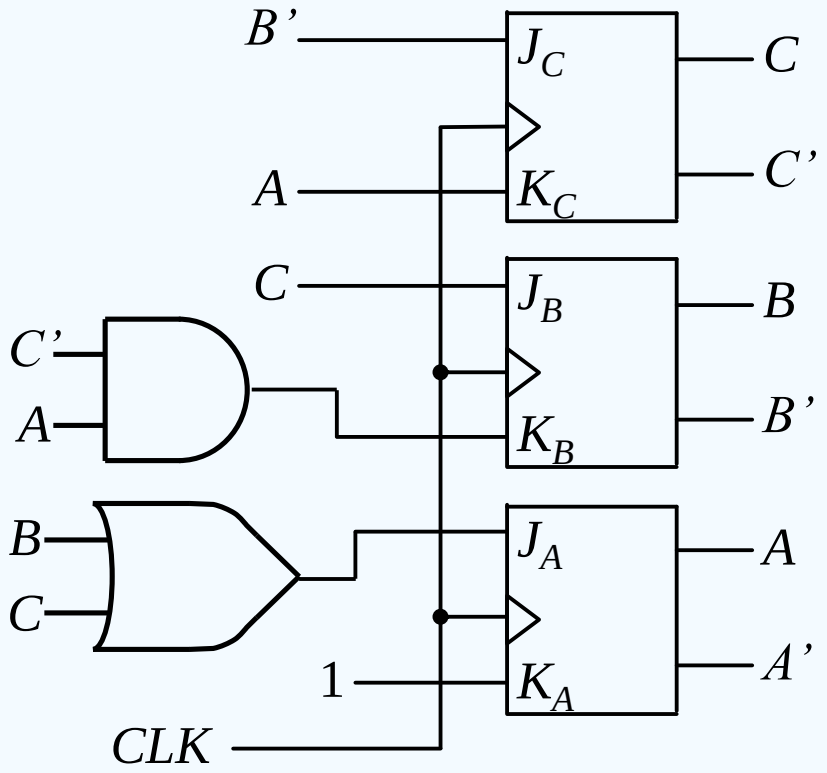
$$J_A = B + C$$
$$K_A = 1$$

Counter Design using J-K Flip-Flops

$$J_C = B'$$
$$K_C = A$$

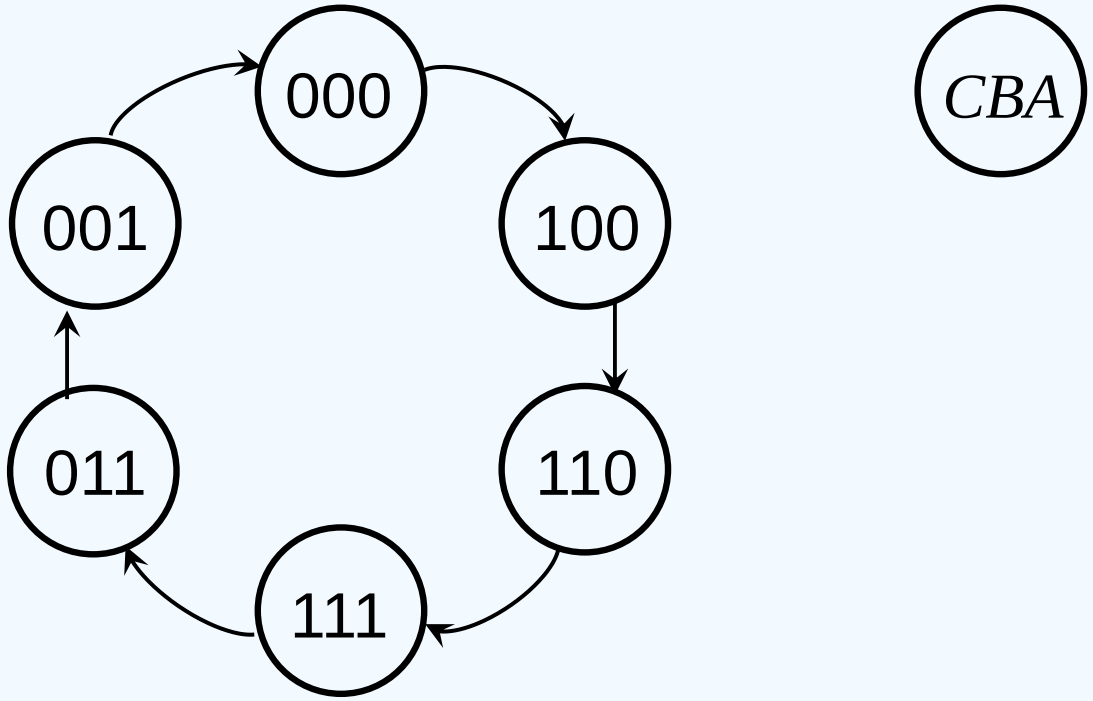
$$J_B = C$$
$$K_B = C'A$$

$$J_A = B + C$$
$$K_A = 1$$



Question

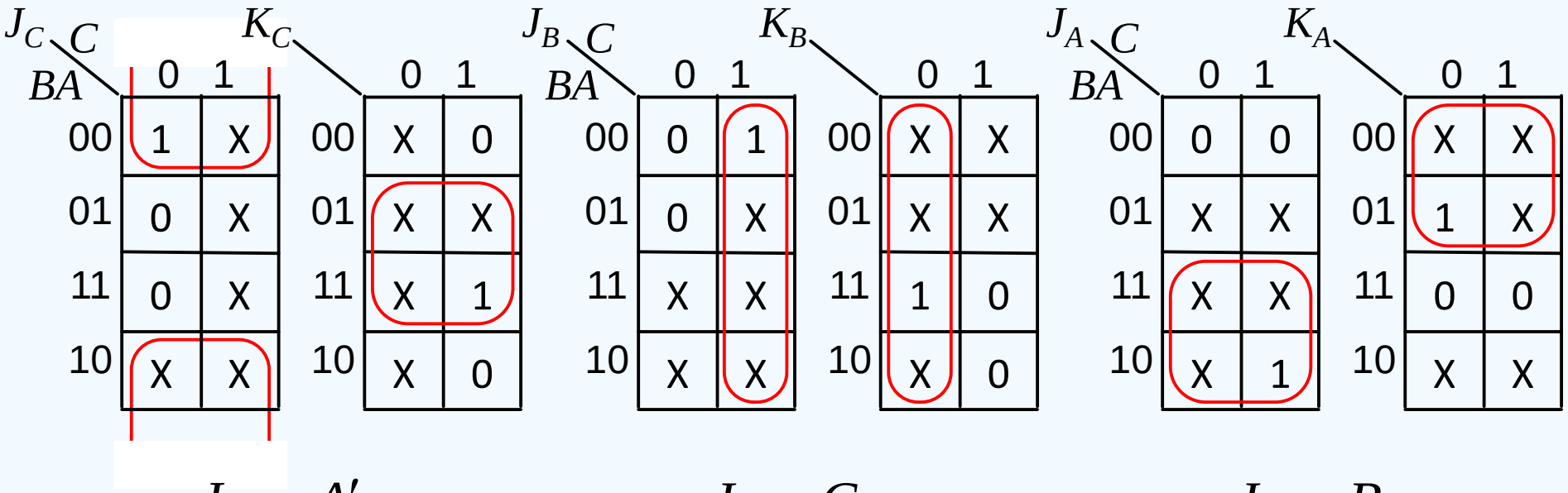
Design a synchronous counter for the following sequence (Gray code) using JK-type flip-flops.



Method

Present State			Next state			Flip-flop inputs					
C	B	A	C^+	B^+	A^+	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	1	0	0	1	X	0	X	0	X
0	0	1	0	0	0	0	X	0	X	X	1
0	1	0	--	--	--	X	X	X	X	X	X
0	1	1	0	0	1	0	X	X	1	X	0
1	0	0	1	1	0	X	0	1	X	0	X
1	0	1	--	--	--	X	X	X	X	X	X
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	1	1	X	1	X	0	X	0

Answer…



$$J_C = A'$$
$$K_C = A$$

$$J_B = C$$
$$K_B = C'$$

$$J_A = B$$
$$K_A = B'$$

(The next-state logic statements we have arrived at here will now allow us to implement this counter design simply, in a similar way to three slides ago…)

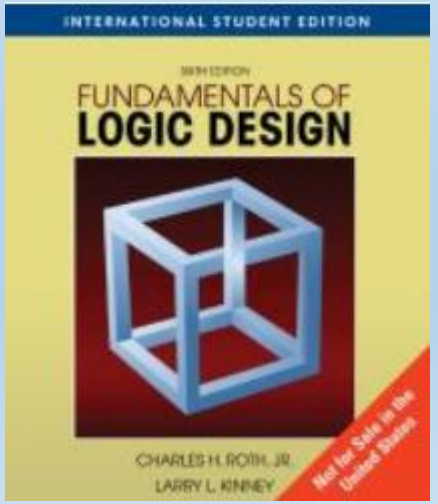
Summary and suggested reading

- Section 11.4 Edge-Triggered D Flip-Flop
- Section 11.7 T Flip-Flop
- Section 11.6 J-K Flip-Flop
- Section 11.8 Additional inputs / Clock Enable
- Chapter 12 Registers and Counters
- Section 12.1 Parallel adder with accumulator

Next lecture:

State machines…

Roth and Kinney *Fundamentals of Logic Design*



UNIVERSITY OF LIVERPOOL