


SEQUENTIAL CIRCUITS - II



©COPYRIGHT CHUA DINGJUAN. ALL RIGHTS RESERVED.

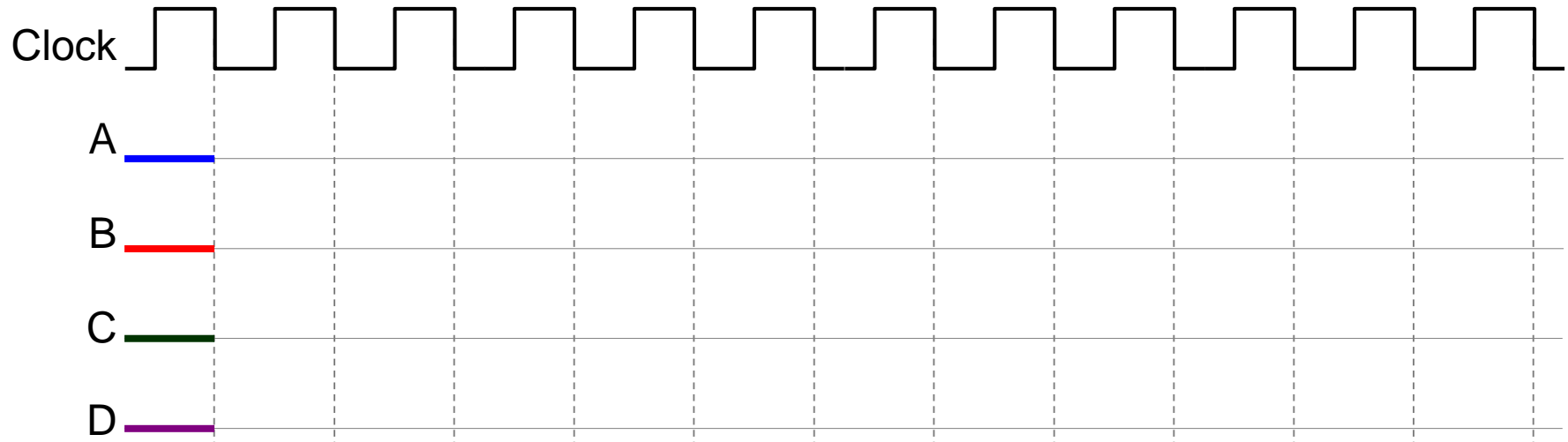
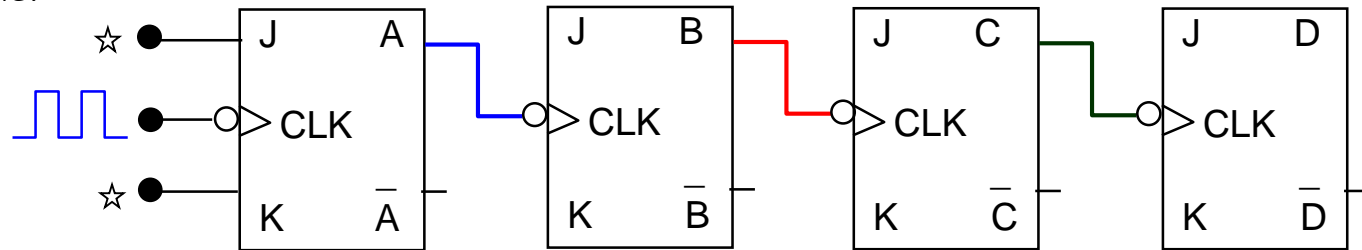
Counters

Asynchronous : Circuit elements do **not** get the clock input simultaneously

Synchronous Counters: Circuit elements get the clock input simultaneously

☆ All J & K inputs HIGH
⇒ FF outputs toggle!

Ripple Counter (Asynchronous):



Counters : Mod - X

- Each FF successively halves the input clock frequency
- The 4-bit counter counts from 0000 (0) → 1111(15)
→ 16 distinct count states ⇒ called **a mod-16 counter**
- N x FFs connected this way will have ____ states ⇒ mod-

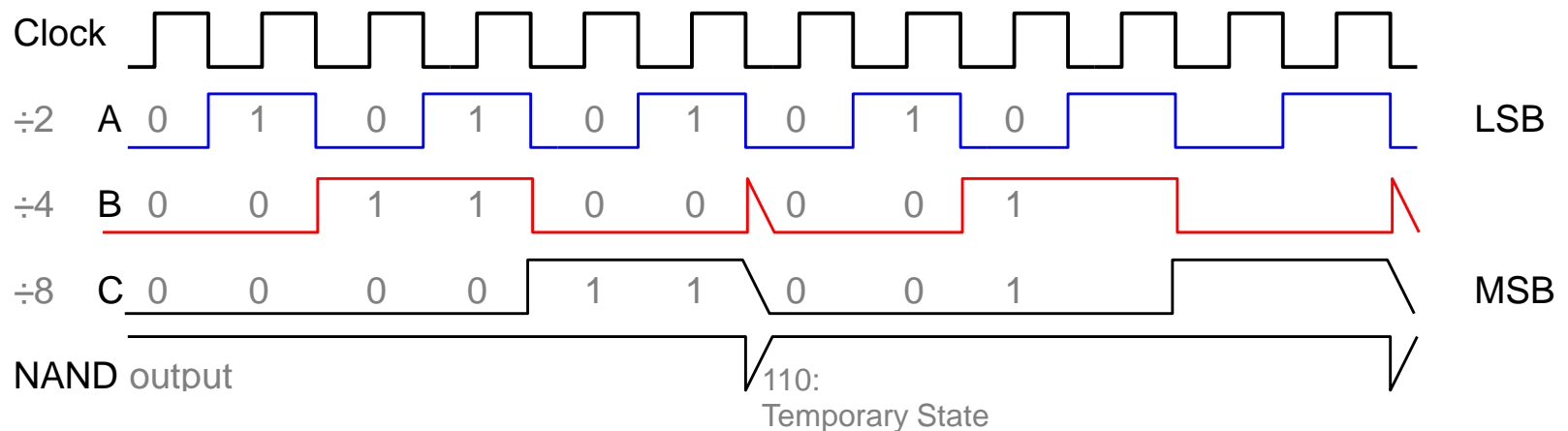
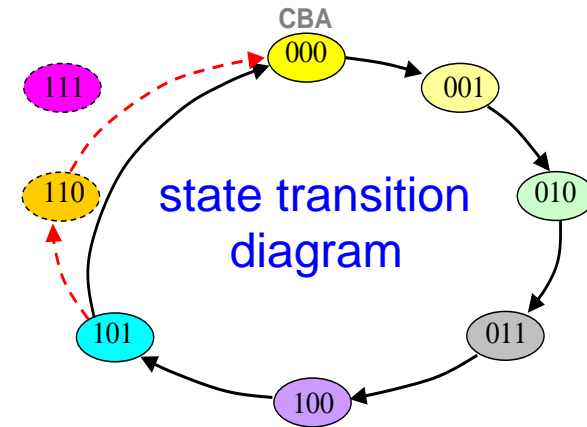
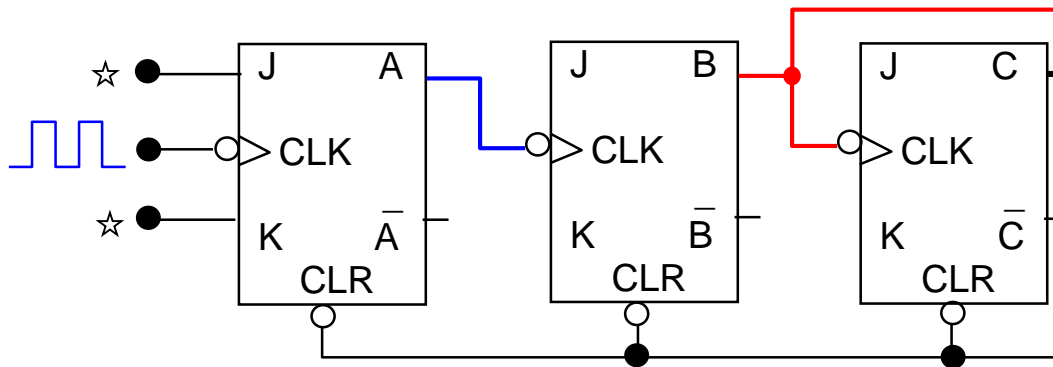
How to obtain a counter with **mod-X** < 2^N ?

1. Assume counter starts from 0
2. Identify FFs that will be in HIGH state when count = X
3. Feed those FFs outputs to a ____ gate
4. Connect ____ gate output to **asynchronous CLR**

CLK	CLR	J	K	Q ⁺
X	L	X	X	L
↓	H	L	L	Q
↓	H	L	H	L
↓	H	H	L	H
↓	H	H	H	\bar{Q}

Mod-6 Asynchronous Counter

1) How many FFs ? _____ 2) Clear at output = ? _____



Mod-? Counters

DC BA

0000

0001

0010

0011

0100

0101

0110

0110

1000

1001

1010

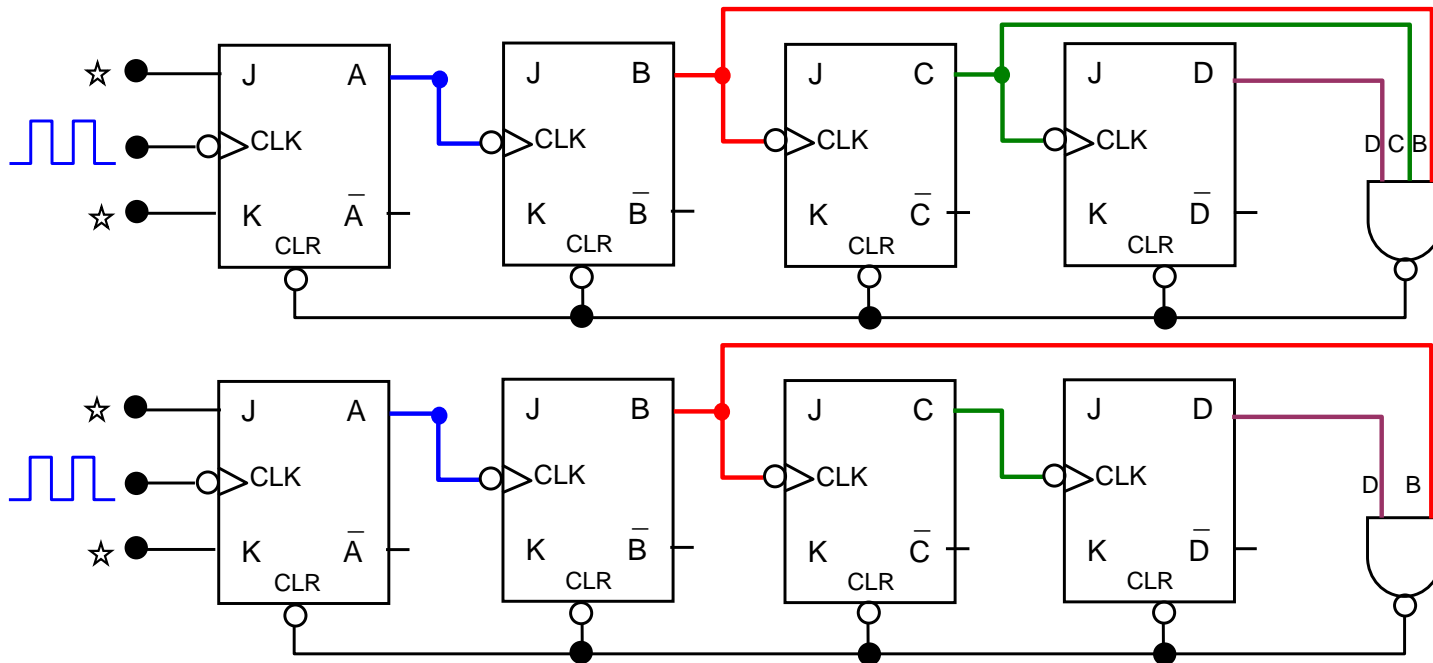
1011

1100

1101

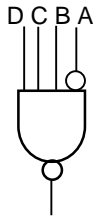
1110

1111



MOD-__ Counter

detects '111X'



MOD-__ Counter

detects '1X1X'



Decade / BCD counter: counts up in ordinary binary sequence

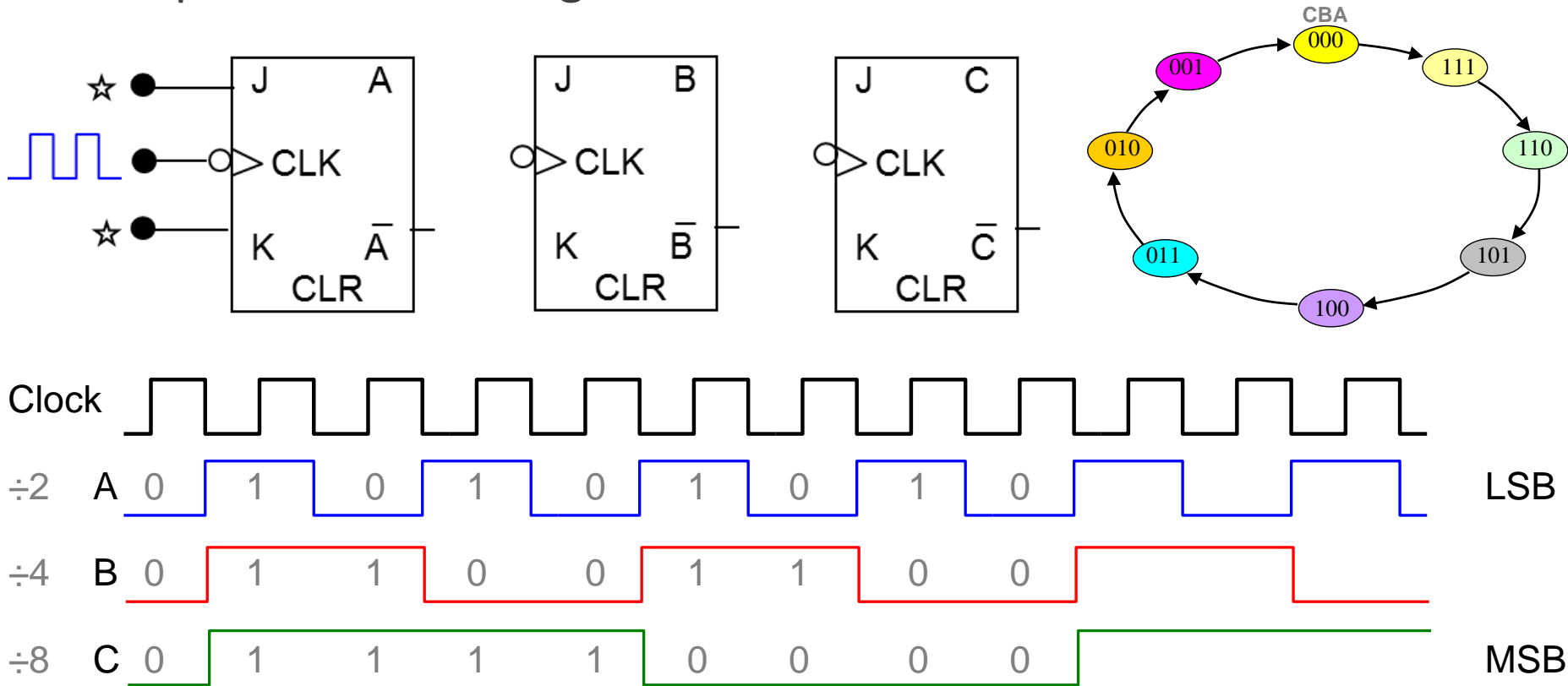
0000 → 1001

Ripple counters considered so far, counted up↑.

How to make them count down↓?

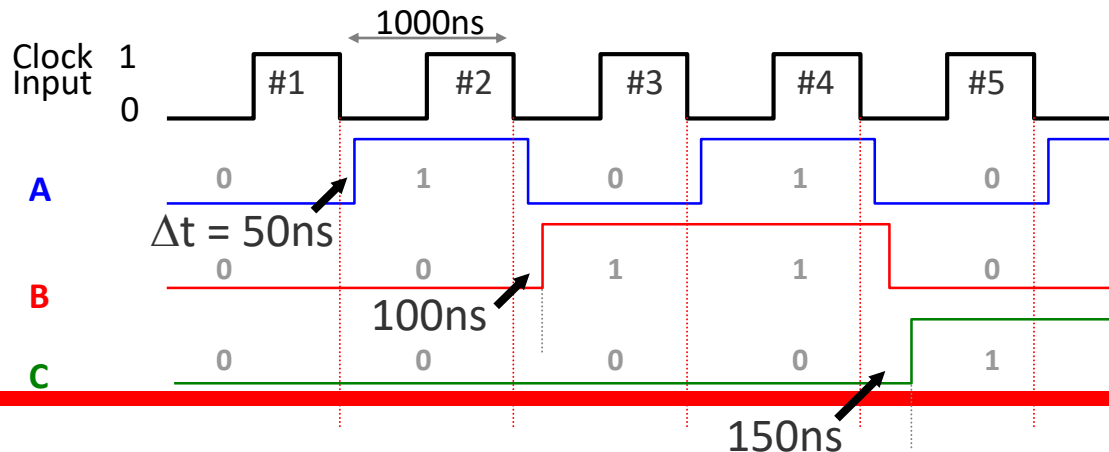
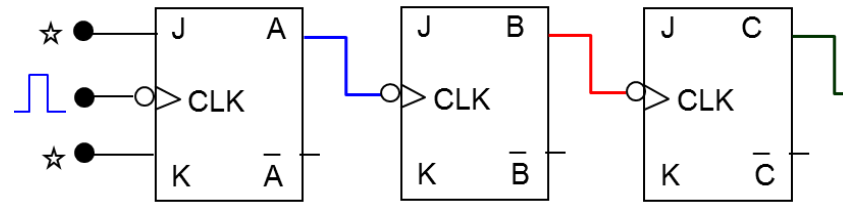
Count Down Ripple Counter...

To count down : connect _____ of FF outputs to clock inputs of succeeding FFs



$t_{pd} \rightarrow$ limiting frequency

- Ripple counters are very easy to implement
- But they have a major drawback \rightarrow they cannot operate beyond a **limiting frequency**
- Due to *propagation delays* of the FFs in the chain adding up:
 1. Clock input FF₁: t_0
 2. Clock input FF₂: $t_0 + \Delta t_{pd}$
 3. Clock input FF₃: $t_0 + 2 * \Delta t_{pd}$
 4. Clock input FF_N: $t_0 + (n - 1) * \Delta t_{pd}$
 \rightarrow the n th FF changes state at $n * \Delta t_{pd}$ after t_0



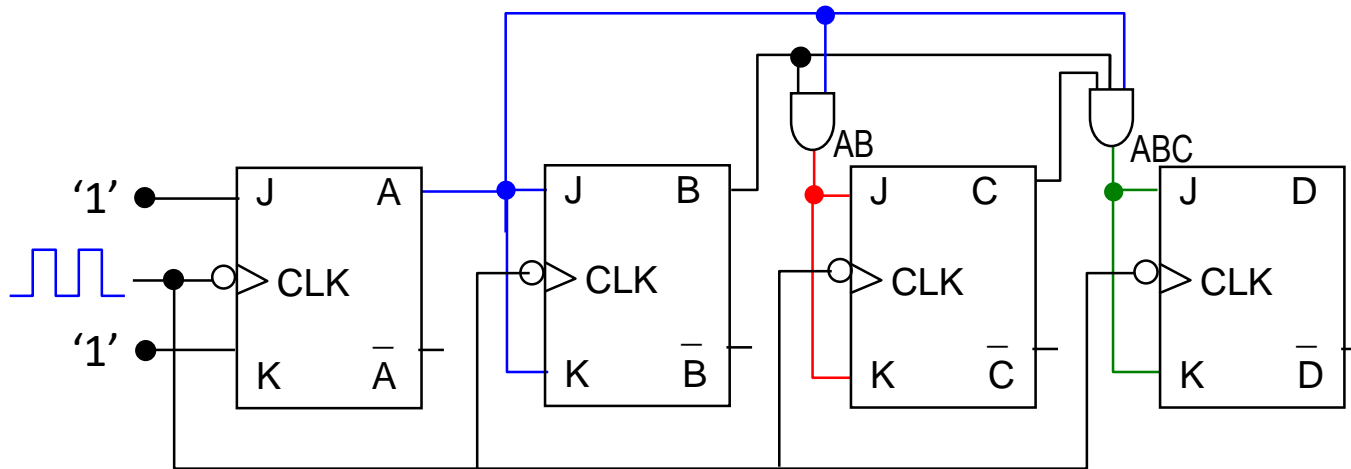
For proper operation:

$$T_{clock} \geq n * \Delta t_{pd}$$

$$f_{max} \leq 1 / n * \Delta t_{pd}$$

Synchronous (Parallel) Counters

Mod-16 Synchronous Parallel Counter :



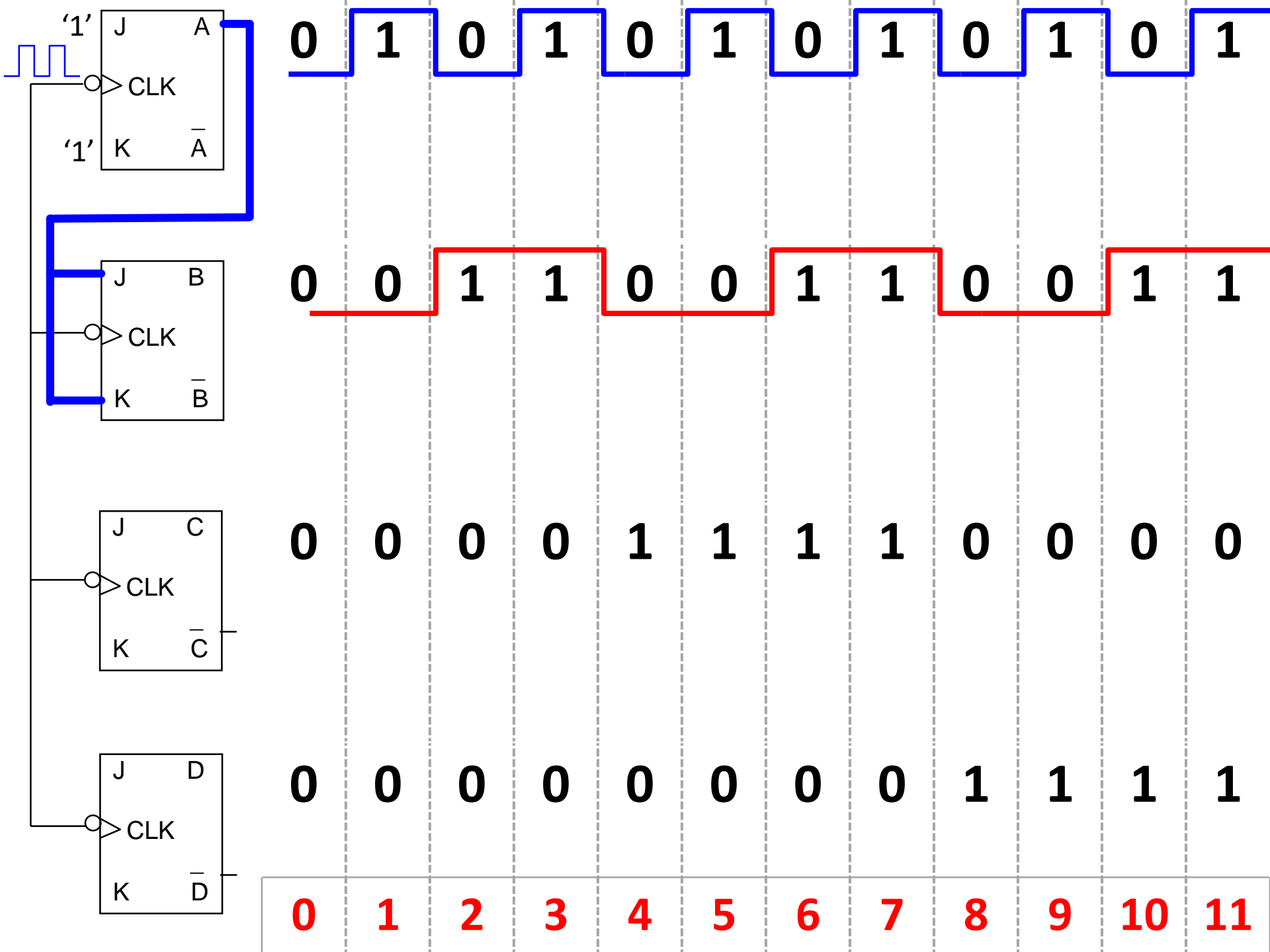
B toggles when A = 1

C toggles when AB = 1

D toggles when ABC = 1

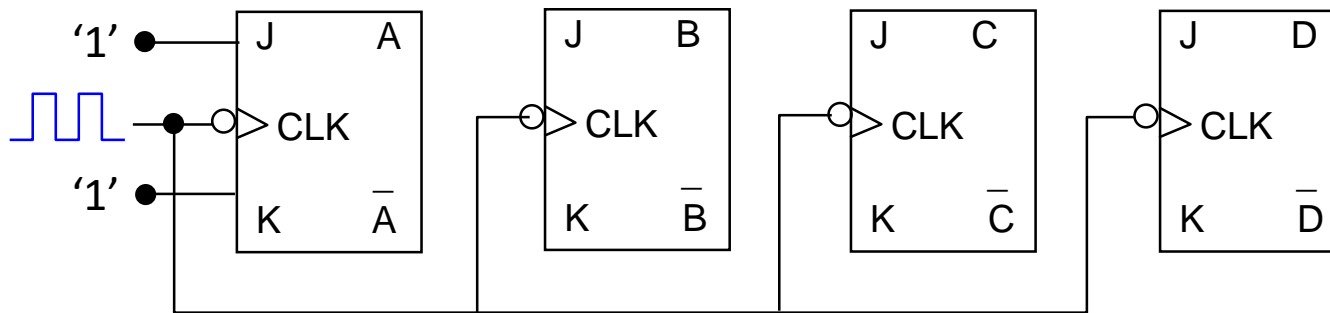
Count	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
0	0	0	0	0
	c	o	n	t

- Since all FFs are triggered simultaneously by the clock, this counter can operate at higher frequencies.
- Total delay = $\Delta t_{pd} \text{ (FF)} + \Delta t_{pd} \text{ (AND)}$
- Operating frequency is irrespective of the number of FFs
- Disadvantages?



Counting Down...

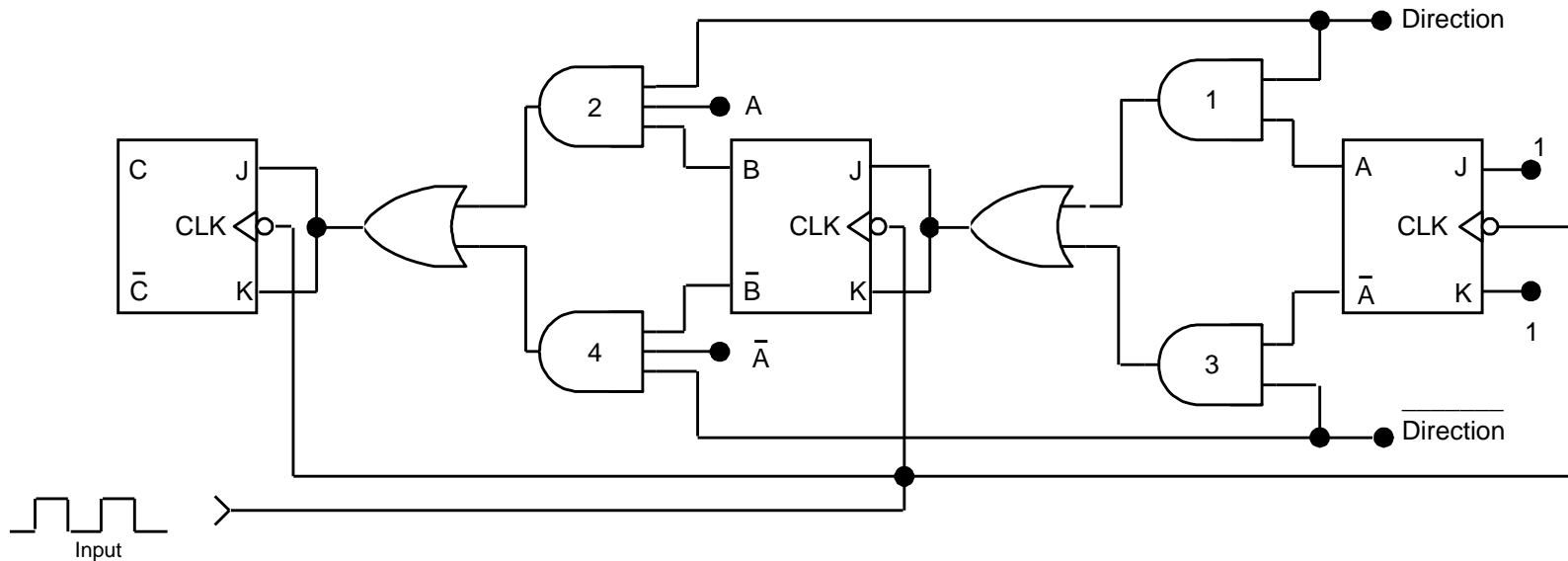
What about a count down mod-16 counter ?



B toggles when
C toggles when
D toggles when

Count	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
0	0	0	0	0
	c	o	n	t

Up/Down Synchronous Counters



○ Counting Up : $Direction = 1, \overline{Direction} = 0$

$J, K_{FFB} =$

$J, K_{FFC} =$

○ Counting Down : $Direction = 0, \overline{Direction} = 1$

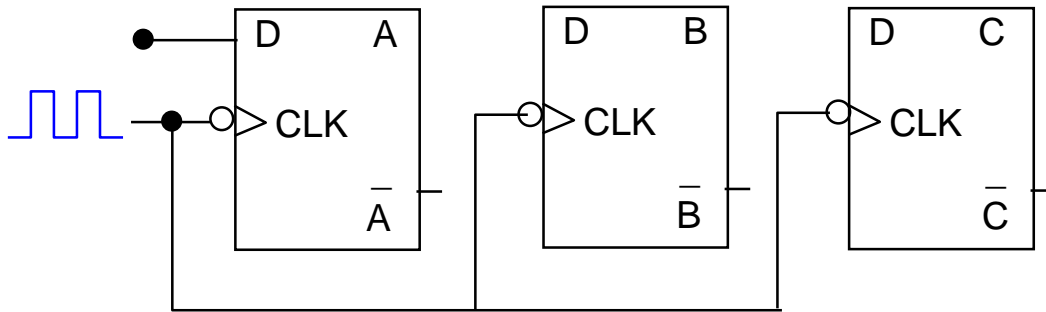
$J, K_{FFB} =$

$J, K_{FFC} =$

Example (D Flip-Flops)

CLK	D	Q ⁺
↑	0	0
↑	1	1

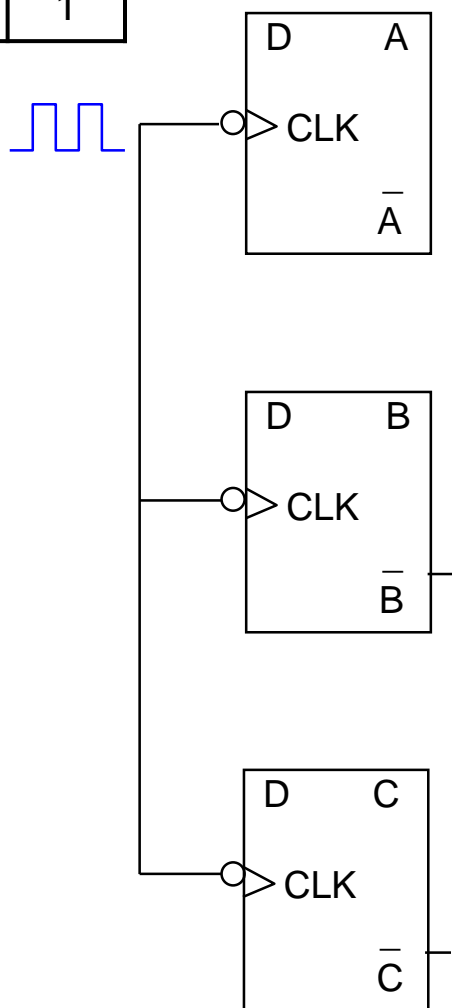
Mod-8 Count-Up Counter Using D-FFs:



Count	C	B	A
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
0	0	0	0
c	o	n	t

CLK	D	Q ⁺
↑	0	0
↑	1	1

Synchronous Counters (DFF)



0 1 0 1 0 1 0 1 0

0 0 1 1 0 0 1 1 0

0 0 0 0 1 1 1 1 0

0 1 2 3 4 5 6 7 0

Verilog!

- Incrementing / Counting is easy in Verilog! → `COUNT <= COUNT + 1;`
- What about the following features?
 - Positive / Negative clock edge triggered
 - Counting Up / Counting Down
 - mod-X Counters
 - Synchronous / Asynchronous Resets
 - Synchronous / Asynchronous Presets

```
module counter(input clear, clk, output reg [3:0] q);
```

```
always @ (posedge clk) begin
```

```
    q <= clear ? (q - 1) : 4'b0000;
```

```
end
```

```
endmodule
```

○ What counter does this code describe?

1. Positive/Negative Edge clock triggered?
2. Asynchronous / Synchronous Clear?
3. Count Up / Down Counter ?