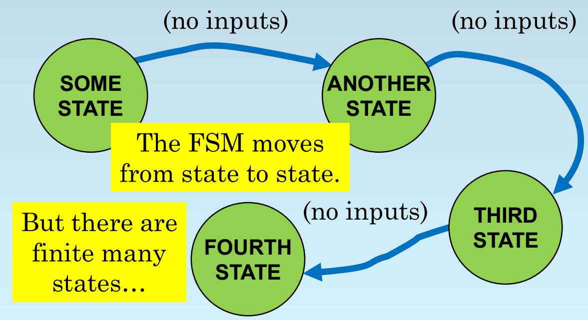
University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering

ECE 120: Introduction to Computing

Binary Counters

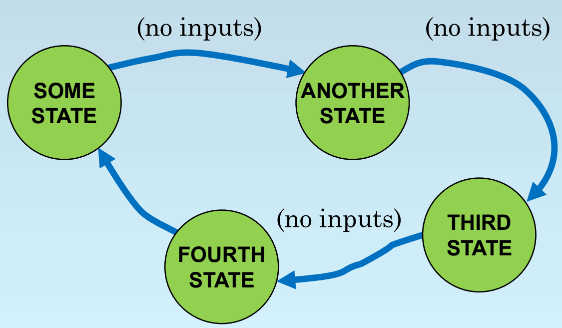
An FSM with No Inputs Moves from State to State

What happens if an FSM has no inputs?



Eventually, the States Form a Loop

So the FSM eventually returns to some state.



A Counter Repeats a Loop of States (Forever)

An FSM without inputs is called a **counter**.

A counter may sometimes have inputs
- to start/stop the counter
- to reset the counter to a known state
- and sometimes to make the counter count in different ways (for example, up or down).

But the basic idea is the same: **the normal operation of a counter is a loop of states**.

We Consider Both Synchronous and Ripple Counters

We focus mainly on

- **synchronous counters**, for which the flip-flops use a common clock signal.
- In other words, they are **clocked synchronous sequential circuits**, and allow us to pretend that time is discrete.

We will also look briefly at

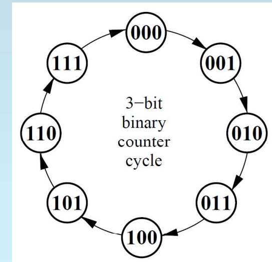
- **ripple counters**, in which flip-flop outputs are used to clock other flip-flops.
- Such designs can save significant power.

Example: 3-Bit Binary Counter

Let's do an example.

The state transition diagram to the right defines a 3-bit binary counter.

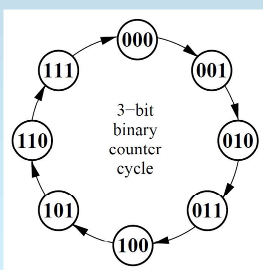
The states correspond to unsigned numbers 0 to 7, after which the counter returns to the 000 state.



Write a Next-State Table

S_2	S_1	S_0	S_2^+	S_1^+	S_0^+
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

Start by writing a next-state table.



Now Use K-Maps to Express the Next-State Values

S_2	S_1	S_0	S_2^+	S_1^+	S_0^+
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

Now copy into K-maps.

$$S_0^+ = S_0' = S_0 \oplus 1$$

S_0^+

S_2 \ S_1S_0	00	01	11	10
0	1	0	0	1
1	1	0	0	1

Now Use K-Maps to Express the Next-State Values

S_2	S_1	S_0	S_2^+	S_1^+	S_0^+
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

Now copy into K-maps.

$$S_1^+ = S_1S_0' + S_1'S_0$$
$$= S_1 \oplus S_0$$

S_1^+

S_2 \ S_1S_0	00	01	11	10
0	0	1	0	1
1	0	1	0	1

Now Use K-Maps to Express the Next-State Values

S_2	S_1	S_0	S_2^+	S_1^+	S_0^+
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

Now copy into K-maps.

$$S_2^+ = S_2'S_1S_0 + S_2S_1'$$
$$+ S_2S_0'$$

S_2^+

S_2 \ S_1S_0	00	01	11	10
0	0	0	1	0
1	1	1	0	1

When Do Place Values Change in Decimal Counting?

When you count in decimal,
when does a place value change?

For example, when does the number of thousands change?

0999 → 1000 1999 → 2000 2999 → 3000

What about the number of ten thousands?

09999 → 10000
19999 → 20000
29999 → 30000

Only **when the lower digits are all 9**.

Can We Use Counting to Generalize the Counter Design?

So **what about in binary**?

Only when the lower digits are all 1.

We have …

$S_0^+ = S_0' = S_0 \oplus 1$

$S_1^+ = S_1S_0' + S_1'S_0 = S_1 \oplus S_0$

$S_2^+ = S_2'S_1S_0 + S_2S_1' + S_2S_0'$

Can you simplify the last equation?

How about $S_2^+ = S_2 \oplus (S_1S_0)$?

Use our General Formula to Build Bigger Counters

If you needed a 4-bit binary counter,
- do you need to draw a K-map?
- Or can you write S_3^+ from our generalization?

$$S_3^+ = S_3 \oplus (S_2 S_1 S_0)$$

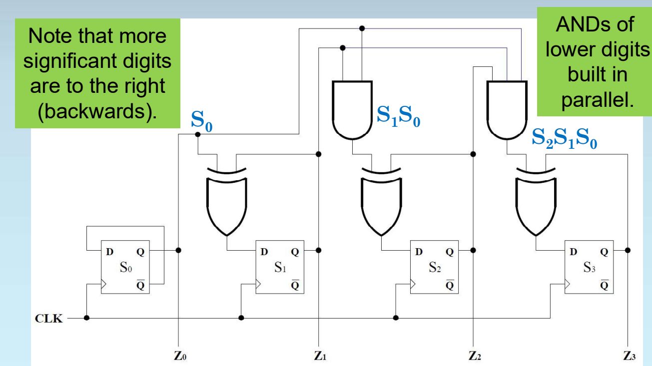
What about S_4^+?

$$S_4^+ = S_4 \oplus (S_3 S_2 S_1 S_0)$$

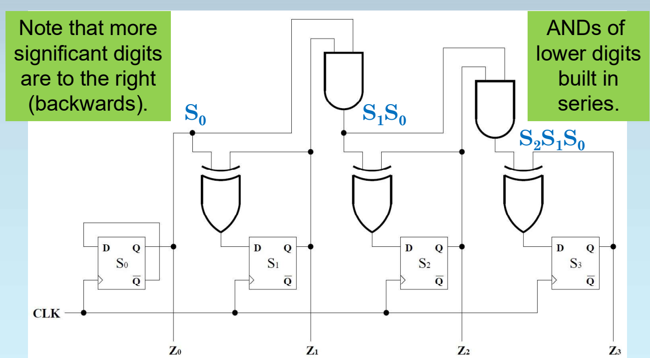
And S_5^+?

$$S_5^+ = S_5 \oplus (S_4 S_3 S_2 S_1 S_0)$$

Binary Counter with Parallel Gating



Binary Counter with Serial Gating



Comparing Serial and Parallel Gating

Parallel gating gives
- bigger gates (more area) and
- less delay.

Serial gating gives
- smaller gates (less area) but
- more delay.

In practice,
- gate sizes are limited, so
- counters use a combination of the two approaches.

Flip-Flops in a Ripple Counter Do Not Share a Clock

We just designed a

synchronous binary counter

flip-flops use a common clock

outputs are binary (unsigned) numbers

one loop of states

Now, let's take a look at a **binary ripple counter**, in which the clock is not shared.

Ripple Counters Require Less Power

In a ripple counter,
- **outputs from some flip-flops**
- are **used to clock other flip-flops** (used as the clock signal input).

Why?
- Recall that changing gate output values implies electric current, which implies power consumption.
- **Clocking flip-flops more slowly reduces energy consumption.**

Ripple Counters are Slower

What's the tradeoff?

Changes to internal state
- ripple through the counter from bit to bit, so
- they are **slower than synchronous counters**.

What about clock skew?

In general, it may be an issue, but
- we will only **consider one simple design**, and
- more complex ripple counters can usually be designed in isolation from other logic.

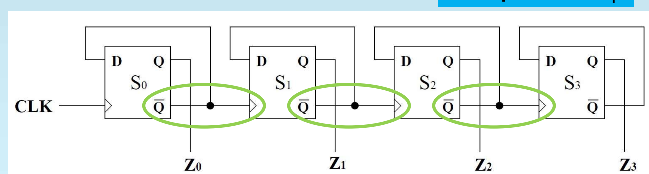
Binary Ripple Counters are Built from Bit Slices

The figure below shows a 4-bit binary ripple counter.

As you can see, the design uses a simple bit slice.

S_i' is the clock input for S_{i+1}.

S_i' is also the D input for S_i.



A Timing Diagram Illustrates Counting