CHAPTER IX

REGISTER BLOCKS
COUNTERS, SHIFT, AND ROTATE REGISTERS

READ PAGES 249-275 FROM MANO AND KIME
Like combinational building blocks, we can also develop some simple building blocks using registers. These include:

- Shift registers
- Rotate registers
- Counters

Implementations of these components can use state machines, but, it is often easier to think of them without the complication of a state machine.
Logical shift registers take the bits stored and move them up a significant bit or down a significant bit.

**LOGICAL SHIFT RIGHT (LSR)**

**ARITHMETIC SHIFT RIGHT (ASR)**

**LOGICAL SHIFT LEFT (LSL)**

**ARITHMETIC SHIFT LEFT (ASL)**

Notice that logical and arithmetic shift lefts are the same.
A simple implementation of a logical right shift register might look like the following.
An arithmetic right shift register might look like the following.
The following is a 4-bit bidirectional shift register with parallel load.
Cascading of shift registers can also be done if the discarded bit is used to shift into another shift register module.

For instance, the 4-bit bidirectional shift register previously presented can be easily cascaded using the

- \( x_r \) (right shift data input) and
- \( x_l \) (left shift data input)
• For example, an 8-bit bidirectional shift register with parallel load can be formed as follows.

\[
\begin{array}{cccccc}
\text{c}_0 & \text{x}_7 & \text{x}_6 & \text{x}_5 & \text{x}_4 & \text{x}_r \\
\text{c}_1 & \text{x}_3 & \text{x}_2 & \text{x}_1 & \text{x}_0 \ \\
\end{array}
\]

\[
\begin{array}{cccc}
\text{z}_3 & \text{z}_2 & \text{z}_1 & \text{z}_0 \\
\end{array}
\]

\[
\begin{array}{cccccc}
\text{x}_3 & \text{x}_2 & \text{x}_1 & \text{x}_0 & \text{x}_r \\
\text{c}_0 & \text{c}_1 & \text{x}_7 & \text{x}_6 & \text{x}_5 & \text{x}_4 \ \\
\end{array}
\]

\[
\begin{array}{cccc}
\text{z}_3 & \text{z}_2 & \text{z}_1 & \text{z}_0 \\
\end{array}
\]
A rotate register is the same as a logical shift register except that the discarded bit is fed back into the empty space from the shift.
• Rotate registers can actually be implemented using shift registers that have serial data inputs (such as the 4-bit bidirectional shift register discussed).

• For example, a 4-bit rotate register can be formed as follows.
A counter is a register that on each clock pulse counts up or down, usually in binary.

Types of counters
- ripple counters
- synchronous counters
- binary counters
- BCD counters
- Gray-code counters
- Ring counters (a 1 moves in a ring from one flip-flop to the next)
- up/down counters (ability to increment or decrement)
- counters with a parallel load (load in starting value with parallel input)
• A modulo-$p$ counter is defined by the following equation.

$$S(t + 1) = (S(t) + x) \mod p$$

• The state diagram for the modulo-$p$ counter is as follows.
• An \( n \)-bit binary counter consists of \( n \) flip-flops and can count in binary from 0 through \( 2^n - 1 \).

• This can be formed with a modulo-\( p \) counter where \( p = 2^n \).

• Two main categories exist for counters:
  • Ripple counters
    • One flip-flop transition serves to trigger other flip-flops.
    • The clock pulse is usually only sent to the first flip-flop.
    • This requires a memory cell that can complement its value.
    • The JK flip-flip would be one approach (we have not studied this!)
  • Synchronous counters
    • Change of state is determined from the present state.
    • Clock pulse sent to all flip-flops.
A toggle cell will be useful for implementing counters.

<table>
<thead>
<tr>
<th>Present Latch Value</th>
<th>CE</th>
<th>CLEAR</th>
<th>Next Latch Value</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
The toggle cell can be used as follows to form a ripple counter.

Notice that the previous toggle cell is connected to the clock input of the next cell. This causes the bits to ripple through the counter.
• Below is an example 4-bit synchronous counter using toggle cells.

- Notice that clock is sent to all toggle cells.
- A simplified form is in Figure 5-11, pp. 269 of Mano & Kime.
Notice that the counters developed so far can count from 0 to $2^n - 1$ for $n$ toggle cells.

- Therefore, for module-$p$ counting, the $p$ is currently limited to $2^n$.
- How about if we wish $p$ to be a non-power of 2?
- Need to build what can be referred to as a divide by counter.
- Given the following counter block, a general modulo-$p$ counter can be constructed by clearing the counter after the desired maximum value.

```
  CE      4-bit counter   \phi_1
    \-----\             \phi_2
    \----\            \-----
CLEAR  \   z_3 \  z_2 \  z_1 \  z_0
          \______\________
```
To illustrate general modulo-\(p\) counters, consider the following implementation of a single digit decimal counter using BCD.

Notice that the counter is cleared after a value of 9 (1001).
The previous BCD counter was built by deriving a terminal count (TC) output signal.

A terminal count output signal for any counter can be useful, so, we will be included in general block diagram for a binary counter.

Notice TC output

In this 4-bit binary counter example, TC=1 only when the output is 1111.
With a terminal count output (TC), counters can be easily cascaded together to form larger counters.

For instance, an 8-bit binary counter can be formed as follows.