Chapter 6

Registers and Counters

Outline

- Registers
- Shift Registers
- Ripple Counters
- Synchronous Counters
- Other Counters
Registers and Counters

- Register:
  - A set of flip-flops, possibly with added combinational gates, that perform data-processing tasks
  - Store and manipulate information in a digital system
- Counter:
  - A register that goes through a predetermined sequence of states
    - A special type of register
  - Employed in circuits to sequence and control operations

The Simplest Register

- Consist of only flip-flops
- Triggered by common clock input
- The Clear input goes to the R (reset) input of all flip-flops
  - Clear = 0 → all flip-flops are reset *asynchronously*
- The Clear input is useful for cleaning the registers to all 0’s prior to its clocked operation
  - Must maintain at logic 1 during normal operations
Register with Parallel Load

- When \( Load = 1 \), all the bits of data inputs are transferred into the registers
  - Parallel loaded
- When \( Load = 0 \), the outputs of the flip-flops are connected to their respective inputs
  - Keep no change

control data loading
at the input side

Outline

- Registers
- Shift Registers
- Ripple Counters
- Synchronous Counters
- Other Counters
The Simplest Shift Register

- Shift register: a register capable of shifting its binary information in one or both directions
- The simplest form: consist of only a chain of flip-flops in cascade

Serial Transfer

- Serial mode: information is transferred and manipulated one bit at a time
- Parallel mode: all the bits of the registers are transferred at the same time
- Shift control: determine when and how many times of the registers are shifted
- Register A is connected in circular mode in this circuit
### Serial Transfer Example

#### Timing Pulse Schedule

<table>
<thead>
<tr>
<th>Timing Pulse</th>
<th>Shift Register A</th>
<th>Shift Register B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial value</td>
<td>1 0 1 1</td>
<td>0 0 1 0</td>
</tr>
<tr>
<td>After T1</td>
<td>1 1 0 1</td>
<td>1 0 0 1</td>
</tr>
<tr>
<td>After T2</td>
<td>1 1 1 0</td>
<td>1 1 0 0</td>
</tr>
<tr>
<td>After T3</td>
<td>0 1 1 1</td>
<td>0 1 1 0</td>
</tr>
<tr>
<td>After T4</td>
<td>1 0 1 1</td>
<td>1 0 1 1</td>
</tr>
</tbody>
</table>

### Serial Adder

- The bits of two binary numbers are added one pair at a time through a single full adder (FA) circuit.
- **Initialization:**
  - $A = \text{augend}$; $B = \text{addend}$
  - Carry flip-flop is cleared to 0
- **For each clock pulse:**
  - A new sum bit is transferred to $A$
  - A new carry is transferred to $Q$
  - Both registers are shifted right
- **To add three or more numbers:**
  - Shift in the next number from the serial input while $B$ is shifted to the FA
  - $A$ will accumulate their sum
Serial v.s. Parallel

- Serial adders:
  - Use shift registers
  - A sequential circuit
  - Require only one FA and a carry flip-flop
  - Slower but require less equipment

- Parallel adders:
  - Use registers with parallel load for sum
  - Basically a pure combinational circuit
  - \( n \) FAs are required
  - Faster

Redesign the Serial Adder

\( J_Q = xy \)
\( K_Q = x'y' = (x + y)' \)
\( S = x \oplus y \oplus Q \)
Universal Shift Register

- The most general shift register has the following capabilities:
  - A clear control to clear the register to 0
  - A clock input to synchronize the operations
  - A shift-right (left) control to enable the shift right (left) operation and the serial input and output lines associated with the shift right (left)
  - A parallel-load control to enable a parallel transfer and the n input lines associated with the parallel transfer
  - n parallel output lines
  - A control state that leaves the information in the register unchanged in the presence of the clock

<table>
<thead>
<tr>
<th>Mode Control</th>
<th>Register Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>S0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Outline

- Registers
- Shift Registers
- **Ripple Counters**
- Synchronous Counters
- Other Counters

Binary Ripple Counter

<table>
<thead>
<tr>
<th>Binary Count Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>A3</td>
</tr>
<tr>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>

The output transition triggers the next flip-flop.
**BCD Ripple Counter**

- The count will return to 0 after 9
- Q1: always complemented
- Q2: inverted when Q8 = 0 and Q1 = 1 \(\rightarrow 0\)
- Q4: inverted when Q2 = 1 \(\rightarrow 0\)
- Q8: when Q1 = 1 \(\rightarrow 0\)
  - if \(Q2 = Q4 = 1\) Q8 is inverted
  - else Q8 = 0

![State Diagram of a Decimal BCD-Counter](image)

**Three-Decade BCD Counter**

connected to the "Count" port

![Block Diagram of a Three-Decade Decimal BCD Counter](image)
Outline

- Registers
- Shift Registers
- Ripple Counters
- Synchronous Counters
- Other Counters

Ripple v.s. Synchronous

- Ripple counters:
  - Flip-flops are triggered by the outputs of another flip-flops
    - Triggering source may not the same for each flip-flop
    - The flip-flops are changed serially

- Synchronous counters:
  - Flip-flops are triggered by common clock pulses
    - Triggering sources are the same for all flip-flops
    - All operations are performed simultaneously
**Binary Counter**

- Triggered by positive clock edge
- Complemented when all the lower bits are "1" (checked by an AND chain)

**J=0,\(K=0\): no change**

**J=1,\(K=1\): complement**

**Up-Down Binary Counter**

- **Up:** 0000→0001→0010→...
- **Down:** 1111→1110→1101→...

Take the complemented values for count-down calculation
Design a BCD Counter

- Go through normal sequential circuit design procedure

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next State</th>
<th>Output</th>
<th>Flip-Flop Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_8$, $Q_4$, $Q_2$, $Q_1$</td>
<td>$Q_8$, $Q_4$, $Q_2$, $Q_1$</td>
<td>$y$</td>
<td>$T_{Q_8}$, $T_{Q_4}$, $T_{Q_2}$, $T_{Q_1}$</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>0 0 1 0</td>
<td>0</td>
<td>0 0 0 1</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>0 0 1 1</td>
<td>0</td>
<td>0 0 0 1</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>0 1 0 0</td>
<td>0</td>
<td>0 1 1 1</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>0 1 0 1</td>
<td>0</td>
<td>0 0 0 1</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>0 1 1 0</td>
<td>0</td>
<td>0 0 1 1</td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>0 1 1 1</td>
<td>0</td>
<td>0 0 0 1</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>1 0 0 0</td>
<td>0</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>1 0 0 1</td>
<td>0</td>
<td>0 0 0 1</td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>0 0 0 0</td>
<td>1</td>
<td>1 0 0 1</td>
</tr>
</tbody>
</table>

$T_{Q_1} = 1$
$T_{Q_4} = Q_2Q_1$
$T_{Q_2} = Q_8Q_1$
$T_{Q_8} = Q_8Q_1 + Q_4Q_2Q_1$

$y = Q_8Q_1$

Binary Counter with Parallel Load

- Controlled by AND gates
- Asynchronous input
- Merged by OR gates

<table>
<thead>
<tr>
<th>Function</th>
<th>Clear</th>
<th>CLK</th>
<th>Load</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear to 0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Load inputs</td>
<td>1</td>
<td>↑</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>Count next</td>
<td>1</td>
<td>↑</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>No change</td>
<td>1</td>
<td>↑</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

set when count = 1111
Achieve a BCD Counter

Load "0000" after "1001"  Clear to "0000" immediately at "1010"

Outline

- Registers
- Shift Registers
- Ripple Counters
- Synchronous Counters
- Other Counters
Counter with Unused States

- If the machine falls into the unused states, we have to bring it back!!
- Cannot assign the used states as don’t cares

\[
\begin{align*}
J_A &= B, \quad K_A = B \\
J_B &= B, \quad K_B = 1 \\
J_C &= B, \quad K_C = 1
\end{align*}
\]

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next State</th>
<th>Flip-Flop Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(a) Logic diagram

Flip-Flop Inputs

The destination can be assigned arbitrarily

Ring Counter

Ring counter: a circular shift register with only one flip-flop being set at any time

(a) Ring-counter (initial value = 1000)

(b) Counter and decoder

(c) Sequence of four timing signals
1. Double the number of states of a ring counter by switch-tail connection.
2. If this counter falls into an unused state, it will never come back to normal states!!
3. To avoid this situation, use some gates to form the input equation of each flip-flop instead of connecting directly.
   - Ex: \( Dc = (A + C) B \)

\[ \begin{array}{c|cccc}
\text{Sequence number} & A & B & C & E \\
\hline
1 & 0 & 0 & 0 & 0 \\
2 & 1 & 0 & 0 & 0 \\
3 & 1 & 1 & 0 & 0 \\
4 & 1 & 1 & 1 & 0 \\
5 & 1 & 1 & 1 & 1 \\
6 & 0 & 1 & 1 & 1 \\
7 & 0 & 0 & 1 & 1 \\
8 & 0 & 0 & 0 & 1 \\
\end{array} \]