UNIT 12
REGISTERS AND COUNTERS

Iris Hui-Ru Jiang Spring 2010

Registers and Counters

Contents
- Register and register transfers
  - Operation
  - Shift register
- Counters
  - Design of binary counters
  - Counters for other sequences
  - Counter design using D FFs
  - Counter design using S-R and J-K
  - Derivation of FF input equations

Reading
- Unit 12

Basic unit
Unit 11: Latch & FFs

Simple sequential Ckt
Unit 12: Registers & Counters

Complex sequential Ckt
Units 13-15: FSM

Put it all together
Unit 16: Summary

Registers (1/2)

- A register: a group of D FFs with a common clock
  - e.g., 4-bit D FF registers with Data, Load, Clear, Clock
  1. Using gated clock
     When Load = 1, load data at \( D \) to \( Q \) at Clk falling

Registers (2/2)

- With clock enable
  When Load = 1, load data at \( D \) to \( Q \) at Clk falling
  When ClrN = 0, clear \( Q \) to 0

Diagram:
- A diagram showing the operation of a 4-bit register with Data, Load, Clear, and Clock inputs.
- The register has four flip-flops labeled \( Q_3, Q_2, Q_1, Q_0 \) with corresponding clear (Clr) and load (Load) signals.
- The clear signal (ClrN) is active low, clearing the register to 0 when ClrN = 0.
- The load signal (Load) is active high, loading data from the input \( D \) to the output \( Q \) when Load = 1.
- The clock (Clk) signal is used to trigger the updates from \( D \) to \( Q \) at the falling edge of Clk.
Data Transfer between Registers

\[ \text{Reg } A = \text{FFs } A_1 \text{ and } A_2 \]
\[ \text{Reg } B = \text{FFs } B_1 \text{ and } B_2 \]
\[ \text{Reg } Q = \text{FFs } Q_1 \text{ and } Q_2 \]

Data transfer btw reg = 2:1 MUX
If \( En = 1 \), \( Q \leftarrow A \); else, \( Q \leftarrow B \)

8-Bit Register with Tri-State Output (1/2)

Symbol

Logic diagram

8-Bit Register with Tri-State Output (2/2)

Data transfer

N-Bit Parallel Adder with Accumulator (1/2)

Adder with accumulator

Adder with MUX

1. Set \( Ld = 1 \); put \( x_i \) on \( y_i \)
2. Set \( Ad = 1 \); put \( y_i \) on \( y_i \)
N-Bit Parallel Adder with Accumulator (2/2)

- **Down to bit-level**
  - 4-bit register/accumulator

```
+-----+-----+-----+-----+
| D   | CE  | D   | CE  |
+-----+-----+-----+-----+
| x_4 | Q^' | x_3 | Q^' |
+-----+-----+-----+-----+
| x_2 | Q   | x_1 | Q   |
+-----+-----+-----+-----+
```

- **Shift Registers (1/2)**
  - A shift register: a group of FFs where binary data can be stored and shifted left or right when a shift signal is applied
  - e.g., 4-bit right-shift register

```
Serial in (SI)

Serial out (SO)

Shift

Clock

Initial, Q_3 Q_2 Q_1 Q_0 = 0101
SI = 1, 1, 0, 1
⇒ Q_3 Q_2 Q_1 Q_0
   0101
   1010
   1101
   0110
   1011
```

- **Shift Registers (2/2)**
  - Timing diagram of a 4-bit right-shift register
N-bit Serial-In Serial-Out Shift Registers

- Take \((n-1)\) cycles to output data

8-bit Serial-In, Serial-Out Shift Register

Parallel-In Parallel-Out (PIPO):
- Load all data at the same time
- Read out data at the same time

4-bit Parallel-In, Parallel-Out Right Shift Register

\[
\begin{array}{cccc}
\text{Sh (Shift)} & \text{L (Load)} & Q_3^+ & Q_2^+ & Q_1^+ & Q_0^+ \\
0 & 0 & Q_3 & Q_2 & Q_1 & Q_0 \\
0 & 1 & D_3 & D_2 & D_1 & D_0 \\
1 & X & SI & Q_3 & Q_2 & Q_1 \\
\end{array}
\]

- Action:
  - No change
  - Load
  - Right shift

PIPO Shift Register Implementation (1/2)

- Implement using FFs and MUXes

Parallel Input

\[
\begin{array}{cccc}
\text{Sh (Shift)} & \text{L (Load)} & Q_3^+ & Q_2^+ & Q_1^+ & Q_0^+ \\
0 & 0 & Q_3 & Q_2 & Q_1 & Q_0 \\
0 & 1 & D_3 & D_2 & D_1 & D_0 \\
1 & X & SI & Q_3 & Q_2 & Q_1 \\
\end{array}
\]

- Action:
  - No change
  - Load
  - Right shift

PIPO (2/2)

- Clock
  - Load
  - Shift

\[
D_3, D_2, D_1, D_0 = 1011
\]

- Action:
  - No change
  - Load
  - Right shift
Shift Register with Inverted Feedback

- **Johnson counter**: a shift register with inverted feedback
- **Counter**: a circuit that cycles through a fixed sequence of states

State graph:

```
Q3 Q2 Q1
011 010 001
```

Counting 0--7 (1/3)

- A **synchronous counter**: FFs are synchronized by a common clock
  - e.g., count 0–7
  1. T FF?

Counting 0--7 (2/3)

- A synchronous counter:
  - FFs are synchronized by a common clock
  - e.g., count 0–7

- **T FF?**

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next State</th>
<th>FF Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
<td>1000</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>1011</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0101</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
<td>1011</td>
</tr>
</tbody>
</table>

Binary Counters

Synchronous counters discussed

- **Johnson counter**: a shift register with inverted feedback
- **Counter**: a circuit that cycles through a fixed sequence of states

State graph:

```
000 001 010 011 100 101 110 111
```

Counting 0--7 (2/3)

- **D FF?**

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
</tr>
</tbody>
</table>

- **T FF?**

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
</tr>
</tbody>
</table>

- **T FF?**

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
</tr>
</tbody>
</table>

- **T FF?**

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
</tr>
</tbody>
</table>

- **T FF?**

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
</tr>
</tbody>
</table>

- **T FF?**

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
</tr>
</tbody>
</table>
Counting $0--7$ (3/3)

Up-Down Counter

State Diagram of a Counter

Counters for Other Sequence
Don’t Care States

- If FFs are initially set to \( CBA = 100 \)
  - Tracking signals through the network shows that \( T_A = T_B = 1 \), so the state changes to 111

- When the power is turned on, the initial states of all FFs are unpredictable!!
  - Don’t care states should be checked to make sure that they eventually lead into the main counting sequence
  - Or use power-up reset

Using D FFs Instead

- \( D_A = A' \cdot (C + B) \)
- \( D_B = B' \cdot (C + B) \)
- \( D_C = C' = B' \)

Recap S-R FFs

- Q: What is the relation between \( S, R \) and \( Q, Q^* \)?

<table>
<thead>
<tr>
<th>( S )</th>
<th>( R )</th>
<th>( Q )</th>
<th>( Q^* )</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Unchanged</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Set ( Q ) to 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Set ( Q ) to 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>- inputs not allowed</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>- inputs not allowed</td>
</tr>
</tbody>
</table>

- However, actually, we do it reversely
  - \( S, R \rightarrow Q, Q^* \)
  - \( S, R \leftarrow Q, Q^* \)
Using S-R FFs (1/3)

Derive S-R FF input maps from the excitation table.

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
</tr>
</thead>
<tbody>
<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>

Using S-R FFs (2/3)

Derive S-R FF inputs from next state maps (faster).

Using S-R FFs (3/3)

Using J-K FFs (1/2)

Registers & counters
Using J-K FFs (2/2)

Derivation of FF Input Equations

Summary

Derivation of Flip-Flop Input Equations

Important Tables

- **Determine the FF input equations from the next-state equations using K-maps**
  - Always copy X's from next state maps onto input maps first
  - Fill in the remaining squares with 0's

<table>
<thead>
<tr>
<th>Type of F/F</th>
<th>Input</th>
<th>Q=0</th>
<th>Q=1</th>
<th>Rules for forming input map from next state map</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>D</td>
<td>0 1</td>
<td>1 0</td>
<td>No change</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>0 1</td>
<td>1 0</td>
<td>No change</td>
</tr>
<tr>
<td>S-R</td>
<td>S</td>
<td>0 1</td>
<td>x</td>
<td>No change</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>x</td>
<td>0 1</td>
<td>Replace 0's with x's</td>
</tr>
<tr>
<td></td>
<td>J-K</td>
<td>J</td>
<td>x</td>
<td>No change</td>
</tr>
<tr>
<td></td>
<td></td>
<td>K</td>
<td>x</td>
<td>Fill in with x's</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type of F/F</th>
<th>Input</th>
<th>Q=0</th>
<th>Q=1</th>
<th>Rules for forming input map from next state map</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>D</td>
<td>0 1</td>
<td>1 0</td>
<td>No change</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>0 1</td>
<td>1 0</td>
<td>No change</td>
</tr>
<tr>
<td>S-R</td>
<td>S</td>
<td>0 1</td>
<td>x</td>
<td>No change</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>x</td>
<td>0 1</td>
<td>Replace 0's with x's</td>
</tr>
<tr>
<td></td>
<td>J-K</td>
<td>J</td>
<td>x</td>
<td>No change</td>
</tr>
<tr>
<td></td>
<td></td>
<td>K</td>
<td>x</td>
<td>Fill in with x's</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type of F/F</th>
<th>Input</th>
<th>Q=0</th>
<th>Q=1</th>
<th>Rules for forming input map from next state map</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>D</td>
<td>0 1</td>
<td>1 0</td>
<td>No change</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>0 1</td>
<td>1 0</td>
<td>No change</td>
</tr>
<tr>
<td>S-R</td>
<td>S</td>
<td>0 1</td>
<td>x</td>
<td>No change</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>x</td>
<td>0 1</td>
<td>Replace 0's with x's</td>
</tr>
<tr>
<td></td>
<td>J-K</td>
<td>J</td>
<td>x</td>
<td>No change</td>
</tr>
<tr>
<td></td>
<td></td>
<td>K</td>
<td>x</td>
<td>Fill in with x's</td>
</tr>
</tbody>
</table>

**Important Tables**

- **Delay**
  - Q Q' | D
  - 0 0 0
  - 0 1 1
  - 1 0 0
  - 1 1 1

- **Toggle**
  - Q Q' | T
  - 0 0 0
  - 0 1 1
  - 1 0 1
  - 1 1 0

- **Set-Reset**
  - Q Q' | S R
  - 0 0 0 0 X
  - 0 1 1 0
  - 1 0 0 1
  - 1 1 X 0

- **Jump-Clear**
  - Q Q' | J K
  - 0 0 0 0 X
Example

Example: 4-Variable Maps (1/3)

Example (2/3)

Example (3/3)
### Loadable Counter

<table>
<thead>
<tr>
<th>Ld</th>
<th>Ctr</th>
<th>ClrN</th>
<th>Ct</th>
<th>Ld</th>
<th>ClrN</th>
<th>Ct</th>
<th>Ld</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>X</td>
<td>Dc</td>
<td>Db</td>
<td>Da</td>
<td>(load)</td>
<td>(no change)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>C</td>
<td>B</td>
<td>A</td>
<td>Present state +1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Ld**: Load
- **Ct**: Count
- **ClrN**: Clear

### Homework for Unit 12

- **Problems**
  - 12.30
  - 12.33
- **Homework #5: Units 12–13**