STANDARD SEQUENTIAL MODULES

- Registers
- Shift registers
- Synchronous counters

For each module we show:
- Its specification
- An implementation with flip-flops and gates
- Its basic uses
- Ways of implementing larger modules with smaller ones
Registers

Figure 11.1: n-bit register module
Register: High-level specification

Inputs: \( \overline{x} = (x_{n-1}, \ldots, x_0), \quad x_i \in \{0, 1\} \)
\( LD, \ CLR \in \{0, 1\} \)

Outputs: \( \overline{z} = (z_{n-1}, \ldots, z_0), \quad z_i \in \{0, 1\} \)

State: \( \overline{s} = (s_{n-1}, \ldots, s_0), \quad s_i \in \{0, 1\} \)

Function: The state transition and output functions are

\[
\overline{s}(t + 1) = \begin{cases} 
  \overline{x}(t) & \text{if } LD(t) = 1 \text{ and } CLR(t) = 0 \\
  \overline{s}(t) & \text{if } LD(t) = 0 \text{ and } CLR(t) = 0 \\
  (0 \ldots 0) & \text{if } CLR(t) = 1 
\end{cases}
\]

\( \overline{z}(t) = \overline{s}(t) \)
Implementation of 4-bit register

Figure 11.2: Implementation of 4-bit register.
Time-behavior of register

Figure 11.3: Time-behavior of a register
Uses of registers: Example 11.1

Input: \( x \in \{0, 1\} \)
Output: \((z_1, z_0), z_i \in \{0, 1\}\)
State: \((s_1, s_0), s_i \in \{0, 1\}\)
Initial state: \((s_1, s_0) = (0, 0)\)

Function: The transition and output functions are:

<table>
<thead>
<tr>
<th>( PS )</th>
<th>( Input )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( x = 0 ) ( x = 1 )</td>
</tr>
<tr>
<td>00</td>
<td>00 01</td>
</tr>
<tr>
<td>01</td>
<td>01 11</td>
</tr>
<tr>
<td>11</td>
<td>11 10</td>
</tr>
<tr>
<td>10</td>
<td>10 00</td>
</tr>
<tr>
<td></td>
<td>( NS )</td>
</tr>
</tbody>
</table>

\[ z(t) = s(t) \]
Canonical implementation

\[ Y_1 = y_1 x' + y_0 x \quad Y_0 = y_0 x' + y'_1 x \]

![Logic diagram](image)

Figure 11.4: Networks for Example 11.1: a) network with state cells;

*Introduction to Digital Systems* 11 – Standard Sequential Modules
Implementation with register

\[ Y_1 = y_0 \quad Y_0 = y_1' \quad LD = x \]

Figure 11.4: Networks for Example 11.1: b) network with standard register module
Shift registers

Figure 11.5: Shift register

Parallel data input \( x \)

Serial data input (right shift) \( x_r \)

Serial data input (left shift) \( x_l \)

Parallel data output \( z \)

CLK
CTRL

[Diagram of a shift register with inputs and outputs labeled]
Parallel-in/parallel-out bidirectional shift register

Figure 11.6: Parallel-in/parallel-out bidirectional shift register
High-level specification

Inputs: \[ x = (x_{n-1}, \ldots, x_0), x_i \in \{0, 1\} \]
\[ x_l, x_r \in \{0, 1\} \]
\[ CTRL \in \{LOAD, LEFT, RIGHT, NONE\} \]

State: \[ s = (s_{n-1}, \ldots, s_0), s_i \in \{0, 1\} \]

Output: \[ z = (z_{n-1}, \ldots, z_0), z_i \in \{0, 1\} \]

Functions: The state transition and output functions:

\[ s(t + 1) = \begin{cases} 
    s(t) & \text{if } CTRL = NONE \\
    x(t) & \text{if } CTRL = LOAD \\
    (s_{n-2}, \ldots, s_0, x_l) & \text{if } CTRL = LEFT \\
    (x_r, s_{n-1}, \ldots, s_1) & \text{if } CTRL = RIGHT 
\end{cases} \]

\[ z = s \]
Shift register control

\[
\begin{array}{c|c}
\text{Control} & s(t + 1) = z(t + 1) \\
\hline
NONE & 0101 \\
LOAD & 1110 \\
LEFT & x_l = 0 \quad 1010 \\
LEFT & x_l = 1 \quad 1011 \\
RIGHT & x_r = 0 \quad 0010 \\
RIGHT & x_r = 1 \quad 1010 \\
\end{array}
\]

\[
\begin{array}{c|cc}
CTRL & c_1 & c_0 \\
\hline
NONE & 0 & 0 \\
LEFT & 0 & 1 \\
RIGHT & 1 & 0 \\
LOAD & 1 & 1 \\
\end{array}
\]
4-bit bidirectional shift-register

Figure 11.7: Implementation of a 4-bit bidirectional shift register using D flip-flops.
Serial-in/serial-out unidirectional shift register

\[ z(t) = x(t - n) \]

Figure 11.8: Common unidirectional shift registers: a) Serial-in/serial-out
Parallel-in/serial-out unidirectional shift register

Figure 11.8: Common unidirectional shift registers: b) Parallel-in/serial-out
Serial-in/parallel-out unidirectional shift register

Figure 11.8: Common unidirectional shift registers: c) Serial-in/parallel-out
Summary of shift-register types

Figure 11.8: Common unidirectional shift registers: a) Serial-in/serial-out; b) Parallel-in/serial-out; c) Serial-in/parallel-out
Uses of shift registers

- serial interconnection of systems

![Diagram of serial interconnection of systems using shift registers](image)

Figure 11.9: Serial interconnection of systems using shift registers

- bit-serial operations
Figure 11.10: Bit-serial adder.
Uses of shift registers: state register

\[ s_{n-1}(t+1) = x(t) \]
\[ s_i(t+1) = s_{i+1}(t) \quad \text{for} \quad i = n - 2, \ldots, 0 \]

- finite-memory sequential system
Example 11.2: shift register as state register

\[ s_7(t + 1) = x(t) \]
\[ s_i(t + 1) = s_{i+1}(t) \quad \text{for } i = 6, \ldots, 0 \]
\[ z(t) = x(t)s_0(t) \]

Figure 11.11: Implementation of network in Example 11.2
Example 11.3: shift register as state register

\[ z(t) = \begin{cases} 
1 & \text{if } s(t) = 01101110 \text{ and } x(t) = 1 \\
0 & \text{otherwise}
\end{cases} \]

Figure 11.12: Implementation of network in Example 11.3
Networks of shift registers

Figure 11.13: Network of serial-input/serial-output shift register modules
• modulo-\( p \) counter

\[
s(t + 1) = (s(t) + x) \mod p
\]

Figure 11.14: State diagram of a modulo-\( p \) counter
A high-level description of a modulo-$p$ counter

Input: \( x \in \{0, 1\} \)

Outputs: \( z \in \{0, 1, \ldots, p - 1\} \)
          \( TC \in \{0, 1\} \)

State: \( s \in \{0, 1, \ldots, p - 1\} \)

Function: The state transition and output functions are

\[
\begin{align*}
    s(t + 1) &= (s(t) + x) \mod p \\
    z(t) &= s(t) \\
    TC(t) &= \begin{cases} 
        1 & \text{if } s(t) = p - 1 \text{ and } x(t) = 1 \\
        0 & \text{otherwise}
    \end{cases}
\end{align*}
\]
Types of counters

- up or down counters

<table>
<thead>
<tr>
<th>State</th>
<th>Binary</th>
<th>BCD</th>
<th>Excess-3</th>
<th>Gray</th>
<th>Ring</th>
<th>Twisted Tail</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>0000</td>
<td>0011</td>
<td>000</td>
<td>00000001</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>0001</td>
<td>0100</td>
<td>001</td>
<td>00000010</td>
<td>0001</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>0010</td>
<td>0101</td>
<td>011</td>
<td>00000100</td>
<td>0011</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>0011</td>
<td>0110</td>
<td>010</td>
<td>00001000</td>
<td>0111</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>0100</td>
<td>0111</td>
<td>110</td>
<td>00100000</td>
<td>1111</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>0101</td>
<td>1000</td>
<td>111</td>
<td>01000000</td>
<td>1110</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>0110</td>
<td>1001</td>
<td>101</td>
<td>01000000</td>
<td>1100</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>0111</td>
<td>1010</td>
<td>100</td>
<td>10000000</td>
<td>1000</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>1011</td>
<td>1011</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>1100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Ring and twisted-tail counters

Figure 11.15: a) Modulo-4 ring counter; b) Modulo-8 twisted-tail counter
Binary counter with parallel input

Inputs: \( I = (I_3, \ldots, I_0), I_j \in \{0, 1\}, I \in \{0, 1, \ldots, 15\} \)
\( CLR, LD, CNT \in \{0, 1\} \)

State: \( s = (s_3, \ldots, s_0), s_j \in \{0, 1\}, s \in \{0, 1, \ldots, 15\} \)

Output: \( s = (s_3, \ldots, s_0), s_j \in \{0, 1\}, s \in \{0, 1, \ldots, 15\} \)
\( TC \in \{0, 1\} \)

Function: The state-transition and output functions are

\[
s(t + 1) = \begin{cases} 
0 & \text{if } CLR = 1 \\
I & \text{if } LD = 1 \\
(s(t) + 1) \mod 16 & \text{if } CNT = 1 \text{ and } LD = 0 \\
s(t) & \text{otherwise}
\end{cases}
\]

\[
TC = \begin{cases} 
1 & \text{if } s(t) = 15 \text{ and } CNT = 1 \\
0 & \text{otherwise}
\end{cases}
\]
Modulo-16 counter

Figure 11.16: A modulo-16 binary counter with parallel input
Modulo-\(k\) counter \((1 \leq k \leq 16)\)

\[
\begin{align*}
CNT & = x \\
LD & = \begin{cases} 
1 & \text{if } (s = k - 1) \text{ and } (x = 1) \\
0 & \text{otherwise}
\end{cases} \\
I & = 0 \\
TC & = LD
\end{align*}
\]
Modulo-$k$ counter ($1 \leq k \leq 16$) (cont.)

Figure 11.17: a) State diagram of modulo-$k$ counter ($1 \leq k \leq 16$); b) Modulo-12 counter and its time behavior ($x = 1$)
a-to-b counter \((0 \leq a, b \leq 15)\)

\[
\begin{align*}
CNT &= x \\
LD &= \begin{cases} 
1 & \text{if } (s = b) \text{ and } (x = 1) \\
0 & \text{otherwise}
\end{cases} \\
I &= a
\end{align*}
\]
Figure 11.18: a) State diagram of an $a$-to-$b$ counter; b) A 1-to-12 counter
Modulo-$k$ frequency divider ($1 \leq k \leq 16$)

\[ CNT = x \]

\[ LD = \begin{cases} 1 & \text{if } TC = 1 \\ 0 & \text{otherwise} \end{cases} \]

\[ I = 16 - k \]

\[ z = TC \]
Figure 11.19: a) State diagram of a modulo-$k$ frequency divider; b) Modulo-9 frequency divider and its time behavior ($x = 1$)
Uses of counters

- count the number of times that a certain event takes place;

- control a fixed sequence of actions

- generate timing signals

- generate clocks of different frequencies

- state register
Sequence of actions:

0: CLEAR ALL REGISTERS
1: INPUT A
2: INPUT B
3: COMPUTE
4: COMPUTE
5: OUTPUT C

Figure 11.20: a) Example of event counter; b) Example of a controller.
Uses of counters (cont.)

Figure 11.21: Examples of networks for generating a) Timing signals; b) Clocks with different frequencies.
Counter as state register

Counting \( s(t + 1) = (s(t) + 1) \mod p \)

No change \( s(t + 1) = s(t) \)

Arbitrary \( s(t + 1) \neq (s(t) + 1) \mod p \) and \( s(t + 1) \neq s(t) \)

Figure 11.22: Implementation of sequential system with counter and combinational networks.
Counter as state register (cont.)

\[ CNT = \begin{cases} 
1 & \text{if } s(t+1) = (s(t) + 1) \mod p \text{ and } x = 1 \\
0 & \text{otherwise} 
\end{cases} \]

\[ LD = \begin{cases} 
1 & \text{if } s(t+1) \neq s(t) \text{ and } \\
 & s(t+1) \neq (s(t) + 1) \mod p \text{ and } x = 1 \\
0 & \text{otherwise} 
\end{cases} \]

\[ I = \begin{cases} 
s(t+1) & \text{if } LD = 1 \\
- & \text{otherwise} 
\end{cases} \]
Example 11.4

Figure 11.23: State diagram for Example 11.4
Example 11.4 (cont.)

\[ CNT = S_0a + S_1 + S_2 + S_3b + S_4c' + S_5 \]

\[ LD = CNT' \]

\[(I_3, I_2, I_1, I_0) = \begin{cases} 
(0, 0, 0, 0) & \text{if } S_0a' + S_6b \\
(0, 0, 0, 1) & \text{if } S_4c + S_6b' \\
(0, 0, 1, 1) & \text{if } S_3b'
\end{cases} \]

Switching expressions for parallel inputs

\[ I_3 = 0 \]
\[ I_2 = 0 \]
\[ I_1 = Q_0 \]
\[ I_0 = Q_0 + Q_2Q'_0 + Q_2b' \]

Output \( z \)

\[ z = Q_1Q_0b' \]
Figure 11.24: Sequential network for Example 11.4
Networks of counters

- Cascade counters

\[ TC = \begin{cases} 
1 & \text{if } (s = p - 1) \text{ and } (CNT = 1) \\
0 & \text{otherwise}
\end{cases} \]

- for the \( i \)-th module

\[ CNT^i = \begin{cases} 
1 & \text{if } (s^{(j)} = p - 1) \text{ and } (x = 1) \\
0 & \text{otherwise}
\end{cases} \]

where \( s^{(j)} \) is the state of counter \( j \).
Cascade counters (cont.)

Figure 11.26: Cascade implementation of a modulo-\(p^k\) counter

- the worst-case delay

\[ T_{\text{worst-case}} = (k - 1)t_{tc} + t_{su} + t_p \]

- the maximum clock frequency possible

\[ f_{\text{max}} = 1/[(k - 1)t_{tc} + t_{su} + t_p] \]
Example 11.5

\[ t_{su} = 4.5[\text{ns}] \] (including the delay of the gates used to produce the inputs to the cells)

\[ t_p = 2[\text{ns}] \]

\[ t_{tc} = 3.5[\text{ns}] \]

Min clock period:

- with one module \( T = 6.5[\text{ns}] \) \( (153[\text{MHz}] \) 
- with 8 modules \( T = 31[\text{ns}] \) \( (32[\text{MHz}] \) 

Introduction to Digital Systems

11 – Standard Sequential Modules
Faster counters

- Introduce $CEF$ (Count Enable First)

$$s(t + 1) = \begin{cases} 
(s(t) + 1) \mod p & \text{if } CEF = 1 \text{ and } CNT = 1 \\
 s(t) & \text{otherwise}
\end{cases}$$

- $TC$ signal not influenced by $CEF$,

$$TC = \begin{cases} 
1 & \text{if } (s(t) = p - 1) \text{ and } (CNT = 1) \\
0 & \text{otherwise}
\end{cases}$$
Faster cascade counter

Figure 11.27: A faster version of a cascade counter
Worst-case delay

\[ T_{\text{worst-case}} = (k - 2)t_{tc} + t_{su} + t_p \]

⇒ small reduction in the delay

* Note: propagation of \( TC \) can span several clock cycles

Consequently, if \( T \) is the clock period,

\[ pT \geq (k - 2)t_{tc} + t_{su} + t_p \]

\[ T \geq t_{tc} + t_{su} + t_p \]

Combining

\[ T \geq \max(t_{tc} + t_{su} + t_p, ((k - 2)t_{tc} + t_{su} + t_p)/p) \]
Figure 11.28: **Timing relations**: a) **Without CEF**; b) **With CEF**
Parallel counters

Modulo-504 counter: $7 \times 8 \times 9$ states

000, 111, 222, 333, 444, 555, 666, 077, 108, 210, 321, 432, ...
Parallel modulo-$(7 \times 8 \times 9)$ counter

Figure 11.29: Parallel implementation of modulo-$(7 \times 8 \times 9)$ counter.
Multimodule systems

- Complex sequential system ⇒ several interacting sequential subsystems

Example 11.6: Block pattern recognizer

- input sequence: blocks of $k$ symbols
- function: check for pattern $P$ in each block
- implementation: counter + recognizer
- output of the counter:

$$TC(t) = \begin{cases} 1 & \text{if } t \mod k = k - 1 \text{ and } \text{check} = 1 \\ 0 & \text{otherwise} \end{cases}$$

- output of the system: $z(t) = p(t) \cdot TC(t)$
Figure 11.30: Block pattern recognizer
Example 11.6: Illustration

<table>
<thead>
<tr>
<th>t</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>5</td>
<td>3</td>
<td>7</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>7</td>
<td>4</td>
<td>3</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>TC</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>p</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>z</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
- count the number of instances of pattern $P$ in blocks of $k$ symbols

Figure 11.30: Block pattern recognizer