PROGRAMMABLE MODULES

- Specification of programmable combinational and sequential modules
  1. PSA
  2. ROM
  3. FPGA
- The way the modules are programmed
- Networks of programmable modules
- Examples of uses
Programmable sequential arrays (PSA)

Figure 12.1: Programmable sequential array (PSA).
Example 12.1: Implementation of sequential systems using PSAs

- Sequence generator

  **Inputs:** $x \in \{0, 1\}$  
  **Outputs:** $z \in \{0, 1, 3, 6, 7, 10, 14\}$

  **Function:** The transition and output functions

  $$
  \begin{align*}
  x = 0 : & \quad z = 0 \rightarrow 10 \rightarrow 14 \rightarrow 7 \rightarrow 0 \cdots \\
  x = 1 : & \quad z = 1 \rightarrow 10 \rightarrow 3 \rightarrow 6 \rightarrow 1 \cdots \\
  x = 0 : & \quad z = 0000 \rightarrow 1010 \rightarrow 1110 \rightarrow 0111 \rightarrow 0000 \cdots \\
  x = 1 : & \quad z = 0001 \rightarrow 1010 \rightarrow 0011 \rightarrow 0110 \rightarrow 0001 \cdots 
  \end{align*}
  $$
Figure 12.2: Timing sequences in Example 12.1.
Example 12.1 (cont.)

<table>
<thead>
<tr>
<th>$y$</th>
<th>$k = 0$</th>
<th>$k = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>$k$</td>
</tr>
<tr>
<td>1</td>
<td>$-$</td>
<td>10 $k$</td>
</tr>
<tr>
<td>3</td>
<td>$-$</td>
<td>6 $x$</td>
</tr>
<tr>
<td>6</td>
<td>$-$</td>
<td>1 $k$</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>$-$ $k$</td>
</tr>
<tr>
<td>10</td>
<td>14</td>
<td>3 $k$</td>
</tr>
<tr>
<td>14</td>
<td>7</td>
<td>$-$ $x$</td>
</tr>
</tbody>
</table>

$y \in \{2, 4, 5, 8, 9, 11, 12, 13, 15\}$ – don’t care states

\[
K = x y_3 y_2 + x y'_3 y'_2 y_1 + k y'_1 + k y'_3 y_2 + k y_3 y'_2
\]
\[
Y_3 = y'_1 + y'_2 k'
\]
\[
Y_2 = y'_3 y'_2 y_1 + y_3 k'
\]
\[
Y_1 = y'_2 + y_3
\]
\[
Y_0 = y_3 k + y_2 k + y_3 y_2
\]
Figure 12.3: PSA implementation in Example 12.1.
Read-only memories (ROM)

Figure 12.4: Read-only memory (ROM)
**Example 12.2**

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x)</td>
<td>(\tilde{z})</td>
</tr>
<tr>
<td>000</td>
<td>1011</td>
</tr>
<tr>
<td>001</td>
<td>1101</td>
</tr>
<tr>
<td>010</td>
<td>0111</td>
</tr>
<tr>
<td>011</td>
<td>1000</td>
</tr>
<tr>
<td>100</td>
<td>0000</td>
</tr>
<tr>
<td>101</td>
<td>1111</td>
</tr>
<tr>
<td>110</td>
<td>1111</td>
</tr>
<tr>
<td>111</td>
<td>1011</td>
</tr>
<tr>
<td>$E$</td>
<td>$x_1$</td>
</tr>
<tr>
<td>-----</td>
<td>------</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

(a) NOR Array

(b) pull-up devices

Figure 12.5: MOS implementation of a $4 \times 4$ read-only memory: a) the function; b) the circuit.
Implementation of switching functions using ROMs

Figure 12.6: ROM-based implementation of a 4-bit adder.
Implementation of sequential systems using ROMs

Inputs: \( x = (x_1, x_0), \ x_i \in \{0, 1\} \)

Outputs: \( z \in \{0, 1, \} \)

State: \( y = (y_1, y_0), \ y_i \in \{0, 1, \} \)

Function: The transition and output function are

\[
\begin{array}{c|ccc}
PS & x_1x_0 \\
\hline
\ y_1y_0 & 01 & 10 & 11 \\
00 & 01,0 & 10,1 & 10,0 \\
01 & 00,0 & 11,1 & 11,0 \\
10 & 11,0 & 10,0 & 00,1 \\
11 & 10,0 & 00,0 & 11,1 \\
\end{array}
\]

\( Y_1Y_0, \ z \)

\( NS, \ Output \)
Figure 12.7: ROM-based implementation of a sequential system: a) network; b) ROM contents.
Types of ROM modules

- Mask-programmed ROM
- Field-programmable ROM (PROMs)
- Erasable ROM (EPROM)
- Electrically erasable ROM (flash-memory) or EEPROM
Networks of programmable modules

\[ f_1(x_4, x_3, x_2, x_1, x_0) = \text{one-set}(0, 3, 11, 12, 16, 23, 27) \]
\[ f_0(x_4, x_3, x_2, x_1, x_0) = \text{one-set}(5, 7, 19, 21, 31) \]

**ROM module:** \( 8 \times 2 \)

\[ \mathbf{x} = (\mathbf{x}^{(0)}, \mathbf{x}^{(1)}) \]
\[ \mathbf{x}^{(0)} = (x_4, x_3) \]
\[ \mathbf{x}^{(1)} = (x_2, x_1, x_0) \]
Figure 12.8: ROM-based network for the implementation of two functions.
Figure 12.9: Implementations of functions with $n$ variables: a) ROMs and decoder; b) ROMs and multiplexer

Introduction to Digital Systems

12 – Programmable Modules
Large number of SFs

Figure 12.10: ROM-based implementation of large number of switching functions.
Field Programmable Gate Arrays (FPGA)

Figure 12.11: Organization of an FPGA chip.
Basic approaches in programming of FPGAs

- On-chip static RAM loaded with configuration bit patterns (SRAM-FPGAs). (volatile)

- Antifuse-programmed devices programmed electrically to provide connections that define chip configuration

- Array-style EPROM and EEPROM programmed devices using several PLAs and a shared interconnect mechanism
Figure 12.12: SRAM FPGA programmable components: (a) Switch. (b) 4-input multiplexer. (c) Look-up table (LUT).
Example: XILINX XC2000

Figure 12.13: A configurable logic block (CLB) (Courtesy of Xilinx, Inc.)
Figure 12.14: SRAM-FPGA options in generating functions: (a) One 4-variable function. (b) Two 3-variable functions. (c) Selection between two functions of 3 variables. (Courtesy of Xilinx, Inc.)
Programmable interconnect

1. Direct interconnections between horizontally and vertically adjacent CLBs—provide fast signal paths between adjacent modules

2. General-purpose interconnect consists of vertical and horizontal wiring segments between switch matrices

3. Long vertical and horizontal lines span the whole CLB array
Figure 12.15: Programmable interconnect. (Courtesy of Xilinx, Inc.)
Example 12.5: BCD adder module

- Implement a one-digit BCD adder using a SRAM-FPGAmodule of XC2000 type

Inputs: \( x = (x_3, x_2, x_1, x_0), \quad x_j \in \{0, 1\}, \quad x \in \{0, \ldots, 9\} \)
\( y = (y_3, y_2, y_1, y_0), \quad y_j \in \{0, 1\}, \quad y \in \{0, \ldots, 9\} \)
\( c_{in} \in \{0, 1\} \)

Outputs: \( s = (s_3, s_2, s_1, s_0), \quad s_j \in \{0, 1\}, \quad s \in \{0, \ldots, 9\} \)
\( c_{out} \in \{0, 1\} \)

Function: \( x + y + c_{in} = 10c_{out} + s \)

- Compute \( 16u + v = x + y + c_{in} \in \{0, \ldots, 19\} \) using a 4-bit binary adder
Example 12.5 (cont.)

- Three cases:

\[
\begin{align*}
\text{Case 1:} & \quad u = 0 \quad v \leq 9 \quad s = v \quad c_{out} = 0 \\
\text{Case 2:} & \quad u = 0 \quad v > 9 \quad s = v - 10 = (v + 6) \mod 16 \quad c_{out} = 1 \\
\text{Case 3:} & \quad u = 1 \quad s = v + 16 - 10 = v + 6 \quad c_{out} = 1
\end{align*}
\]

\Rightarrow \text{BCD output}

\[
s = \begin{cases} 
(v + 6) \mod 16 & \text{if } u = 1 \text{ or } v \geq 10 \\
v & \text{otherwise}
\end{cases}
\]

\[
c_{out} = \begin{cases} 
1 & \text{if } u = 1 \text{ or } v \geq 10 \\
0 & \text{otherwise}
\end{cases}
\]

The condition \( u = 1 \text{ or } v \geq 10 \) corresponds to switching expression

\[
t = u + v_3v_2 + v_3v_1
\]
Example 12.5 (cont.)

Figure 12.16: Implementation of BCD adder module
Example 12.5 (cont.)

- Simplification of the 3-bit adder

\[
\begin{align*}
    s_3 &= v_3 \oplus t(v_2 \oplus v_1) \\
    s_2 &= v_2 \oplus tv_1' \\
    s_1 &= v_1 \oplus t
\end{align*}
\]

Moreover,

\[
\begin{align*}
    s_0 &= v_0 \\
    c_{out} &= t
\end{align*}
\]
Design with FPGAs

- Involves intensive use of CAD tools and module libraries

**Design entry**: a schematic entry or a behavioral description

**Implementation**: 
- partition of design into submodules that can be mapped onto CLBs,
- placement of submodules onto chip, and
- routing of signals to connect the submodules

**Design verification**: 
- in-circuit testing
- simulation, and
- timing analysis