ELE2120 Digital Circuits and Systems

Tutorial Note 8
Outline

1. Register
2. Counters
3. Synchronous Counter
4. Asynchronous Counter
5. Sequential Circuit Design
Overview

1. Register
   – Applications:
     • temporally store bits
     • Shifting bits for calculation, serial comm.
   – Shift register, serial adder, USR.

2. Counter
   – Asynchronous Counters
   – Synchronous Counters

3. Clock pulse generator
Shift Register

- Array of Flip-Flops
- is a synchronous systems
  – they are driven by the same clocking waveform
Parallel-Access Shift Register
Exercise (1): Register

- An application of shift registers is in the generation of pseudo-noise sequences for mobile communications. The 4 registers below shift to the right at each clock pulse. Let the initial contents of the registers be 0001. Give the output sequence for 20 clock pulses. Is the output periodic? If it is, what is the period? How does the initial contents of the registers affect the output sequence?
Exercise(1): Register

<table>
<thead>
<tr>
<th>Fth Pulse</th>
<th>State of Register</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- The output is periodic with period equals to 6
Counters

• Also an array of Flip Flops
• Changes of state depends on the changes of “event”
  – Clock driven – synchronous
  – Event driven – asynchronous
• Concept of modulo (mod)
  – Max. no. of states = 2^n states (with n arrays)
  – no. of states through the cycle before turning back to the starting states (mod)
Synchronous Counter

- Ripple Counters
- Event Driven
- Possible of saving hardware
- Delay
3-bit Up Counter

- 3-bit Up counter with T F-F

• Timing diagram
4-bit Up Counter

• 4-bit Up counter with T F-F

Overflow Conditions: only when Q0=1&Q1=1&Clk=1.
Synchronous Counter

When Load = 0:

\[
\begin{align*}
D_0 &= Q_0 \oplus \text{Enable} \\
D_1 &= Q_1 \oplus Q_0 \cdot \text{Enable} \\
D_2 &= Q_2 \oplus Q_1 \cdot Q_0 \cdot \text{Enable} \\
D_3 &= Q_3 \oplus Q_2 \cdot Q_1 \cdot Q_0 \cdot \text{Enable}
\end{align*}
\]

When Load = 1:

D0,D1,D2,D3 will be assign to Q0,Q1,Q2,Q3
Synchronous Counter

- A Modulo-6 Counter with S Reset

Detect and reset when 5 occurs
Asynchronous Counter (binary & BCD)

2-Digit BCD Counter

Four parts to form an AS counter:

1. Enable: $D_i$ with respect of ‘$Q_i$’ and ‘Enable’.

2. Load: Reset $D_0, D_1, D_2, D_3$ when ‘Load’ equals to 1.

3. Clear: Load 0’s into the counter.

4. Overflow carrier: Connect with ‘Load’ to check the reset condition.
Synchronous Counter

• Ring Counter
• Share the same clocking system
• mod 4 counter
Exercise(1)

- Draw the state table and state diagram of the sequential circuit shown. Is there a redundant state? Explain.
According to the system, “11” state should be redundancy as the circuit will count with the loop contains “00”, “01”, “10” only except if the circuit was set initially at “11” before starting.
Exercise (2)

Q: A sequential circuit with two T-FF, A & B, and an input x has the following input equations for the FF:

\[ T_A = Bx + A' \]
\[ T_B = Ax + B \]

1. Draw the circuit diagram;
2. Draw the state table;
3. Draw the state state diagram.
Exercise (2)

- **Circuit Diagram:**

- **State Diagram:**

- **State Table:**

<table>
<thead>
<tr>
<th>Present state</th>
<th>Input</th>
<th>FF input</th>
<th>Next state</th>
</tr>
</thead>
<tbody>
<tr>
<td>A  B  x  T_A  T_B</td>
<td>A  B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0  0  0  1  0</td>
<td>1  0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0  0  1  1  0</td>
<td>1  0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0  1  0  1  1</td>
<td>1  0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0  1  1  1  1</td>
<td>1  0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1  0  0  0  0</td>
<td>1  0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1  0  1  0  1</td>
<td>1  1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1  1  0  0  1</td>
<td>1  0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1  1  1  1  1</td>
<td>0  0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sequential Circuit Design

• In sequential circuit design, we turn some description into a working circuit.
  – We first make a state table or diagram to express the computation.
  – Then we can turn that table or diagram into a sequential circuit.

• Need to be clear:
  – Characteristic table and characteristic equations are referring to latches and Flip Flops.
  – State equations, state tables and state diagrams are talking about a particular design.
Sequential Circuit Design

T flip-flops lead to a simpler circuit.

D flip-flops have the advantage that you don’t have to set up flip-flop inputs at all, since $Q(t+1) = D$. However, the D input equations are usually more complex than T input equations.

In practice, D flip-flops are used more often.
– There is only one input for each flip-flop, not two.
– There are no excitation tables to worry about.
Sequential Circuit Design

Sequential circuit design procedure

Step 1:
Make a state table based on the problem statement. The table should show the present states, inputs, next states and outputs. (It may be easier to find a state diagram first, and then convert that to a table.) Do state reduction if possible.

Step 2:
Assign binary codes to the states in the state table, if you haven’t already. If you have $n$ states, your binary codes will have at least $\lceil \log_2 n \rceil$ digits, and your circuit will have at least $\lceil \log_2 n \rceil$ flip-flops.

Step 3:
For each flip-flop and each row of your state table, find the flip-flop input values that are needed to generate the next state from the present state. You can use flip-flop excitation tables here.

Step 4:
Find simplified equations for the flip-flop inputs and the outputs.

Step 5:
Build the circuit!
Exercise

Design a BCD counter that will count from 0000(zero) through 1001(decima 9).
To construct a counter that starts counting from all 0s to X:

1. Find the smallest number of FFs such that $2^N \geq X$, and connect them as a counter. If $2^N = X$, do not do step 2 and 3.

2. Connect a NAND gate to the asynchronous Clear inputs of all the FFs.

3. Determine which FFs will be in the HIGH state at a count = X; then connect the normal outputs of these FFs to the NAND gate inputs.
Solution

$2^3 = 8$, $2^4 = 16$; thus four FFs are required. Since the counter is to have stable operation up to the count of 1001, it must be reset to zero when the count of 1010 is reached. Therefore, FF outputs D and B must be connected as the NAND gate inputs.
Exercise

Determine \( Q(t+\Delta t) \) as a function of the inputs and \( Q(t) \) for the logic circuit shown in Figures:
Typical question based on characteristic equation.
1. Find the next-state equation of the same kind.

For this case, it is a gated SR latch, the next-state equation:

\[ Q(t+\Delta t) = S(t)C(t) + R'(t)Q(t) + C'(t)Q(t) \]

2. Identify the input and output

\[ S(t) = (X(t)+Y(t))' = X'(t)Y'(t) \]
\[ R(t) = Y(t) \]

3. Substitute the input and output to it

\[ C = 1, \]
\[ Q(t+\Delta t) = X'(t)Y'(t) + Y'(t)Q(t) \]
Ref. to Characteristic Equation

• Specify next state as a function of its current state and inputs

• Q(t) ➞ current state
• Q(t+1) ➞ next state

• For example:
• SR latch: Q(t+1) = S + R’Q(t)
• D flip-flop: Q(t+1) = D
• JK flip-flop: Q(t+1) = JQ’(t)+K’Q(t)
• T flip-flop: Q(t+1) = T ⊕ Q(t)= TQ’(t)+T’Q(t)