ELEN0040 - Electronique numérique

(Patricia ROUSSEAU)

IMPROVED BY FRED SENNY

Année académique 2014-2015
CHAPITRE 4

Sequential circuits
1 Fundamentals of Sequential Circuits
   1.1 Motivation
   1.2 Synchronous and Asynchronous Circuits
   1.3 State, State Diagram and State Table
   1.4 Time simulation

2 Latches

3 Flip-Flops

4 State diagrams and State Tables

5 Finite State Machine Diagrams
Real Basic Memory Element (1bit) : the LATCH ("Verrou")

- state = 1 binary variable = 1 bit
- capability to force output to 0 or 1
- asynchronous storage elements
- from basic to more elaborated latches:
  1. basic (NOR) $SR$ Latch
  2. basic (NAND) $\bar{S}\bar{R}$ Latch
  3. clocked $SR$ Latches
  4. $D$ Latch
Basic SR Latch

- Formed by 2 cross-coupled NOR gates
- The states are defined by outputs $Q$, $\bar{Q}$ which normally are reciprocally complemented values
- There are thus 2 useful states:
  - the Set State: $Q = 1$, $\bar{Q} = 0$
  - the Reset State: $Q = 0$, $\bar{Q} = 1$
- There are two inputs:
  - set input $S : S = 1$ brings the system in its Set state
  - reset input $R : R = 1$ brings the system in its Reset state
SR Latch behavior

1. start with $R = 0$, $S = 0$, the stored state is initially unknown
2. $S$ changes to 1, this sets $Q$ to 1
3. $S$ back to 0, $Q$ “remembers” 1 thus, two input conditions cause the system to be in set state
4. $R$ changes to 1, this resets $Q$ to 0
5. $R$ back to 0, now $Q$ “remembers” 0 thus, two input conditions cause the system to be in reset state
6. suppose both $S$ and $R$ changes to 1
7. both $Q$ and $\overline{Q}$ are zero !!!, undefined state
8. if both $R$ and $S$ go to zero simultaneously, can lead to unstable or “race” condition, oscillating between 00 and 11 undefined states
SR Latch behavior

1. start with $R = 0$, $S = 0$, the stored state is initially unknown
2. $S$ changes to 1, this sets $Q$ to 1
3. $S$ back to 0, $Q$ "remembers" 1 thus, two input conditions cause the system to be in set state
4. $R$ changes to 1, this resets $Q$ to 0
5. $R$ back to 0, now $Q$ "remembers" 0 thus, two input conditions cause the system to be in reset state
6. suppose both $S$ and $R$ changes to 1
7. both $Q$ and $\overline{Q}$ are zero !!!, undefined state
8. if both $R$ and $S$ go to zero simultaneously, can lead to unstable or "race" condition, oscillating between 00 and 11 undefined states
SR Latch behavior

1. start with $R = 0$, $S = 0$, the stored state is initially unknown
2. $S$ changes to 1, this sets $Q$ to 1
3. $S$ back to 0, $Q$ “remembers” 1 thus, two input conditions cause the system to be in set state
4. $R$ changes to 1, this resets $Q$ to 0
5. $R$ back to 0, now $Q$ “remembers” 0 thus, two input conditions cause the system to be in reset state
6. suppose both $S$ and $R$ changes to 1
7. both $Q$ and $\bar{Q}$ are zero!!!, undefined state
8. if both $R$ and $S$ go to zero simultaneously, can lead to unstable or “race” condition, oscillating between 00 and 11 undefined states
SR Latch behavior

1. start with \( R = 0, S = 0 \), the stored state is initially unknown

2. \( S \) changes to 1, this sets \( Q \) to 1

3. \( S \) back to 0, \( Q \) “remembers” 1 thus, two input conditions cause the system to be in set state

4. \( R \) changes to 1, this resets \( Q \) to 0

5. \( R \) back to 0, now \( Q \) “remembers” 0 thus, two input conditions cause the system to be in reset state

6. suppose both \( S \) and \( R \) changes to 1

7. both \( Q \) and \( \bar{Q} \) are zero !!!, undefined state

8. if both \( R \) and \( S \) go to zero simultaneously, can lead to unstable or “race” condition, oscillating between 00 and 11 undefined states
SR Latch behavior

1. start with $R = 0$, $S = 0$, the stored state is initially unknown
2. $S$ changes to 1, this sets $Q$ to 1
3. $S$ back to 0, $Q$ “remembers” 1 thus, two input conditions cause the system to be in set state
4. $R$ changes to 1, this resets $Q$ to 0
5. $R$ back to 0, now $Q$ “remembers” 0 thus, two input conditions cause the system to be in reset state
6. suppose both $S$ and $R$ changes to 1
7. both $Q$ and $\overline{Q}$ are zero !!!, undefined state
8. if both $R$ and $S$ go to zero simultaneously, can lead to unstable or “race” condition, oscillating between 00 and 11 undefined states
SR Latch behavior

1. start with $R = 0$, $S = 0$, the stored state is initially unknown
2. $S$ changes to 1, this sets $Q$ to 1
3. $S$ back to 0, $Q$ “remembers” 1 thus, two input conditions cause the system to be in set state
4. $R$ changes to 1, this resets $Q$ to 0
5. $R$ back to 0, now $Q$ “remembers” 0 thus, two input conditions cause the system to be in reset state
6. suppose both $S$ and $R$ changes to 1
7. both $Q$ and $\overline{Q}$ are zero !!!, undefined state
8. if both $R$ and $S$ go to zero simultaneously, can lead to unstable or “race” condition, oscillating between 00 and 11 undefined states
SR Latch behavior

1. start with \( R = 0, S = 0 \), the stored state is initially unknown
2. \( S \) changes to 1, this sets \( Q \) to 1
3. \( S \) back to 0, \( Q \) “remembers” 1 thus, two input conditions cause the system to be in set state
4. \( R \) changes to 1, this resets \( Q \) to 0
5. \( R \) back to 0, now \( Q \) “remembers” 0 thus, two input conditions cause the system to be in reset state

6. suppose both \( S \) and \( R \) changes to 1
7. both \( Q \) and \( \overline{Q} \) are zero !!!, undefined state
8. if both \( R \) and \( S \) go to zero simultaneously, can lead to unstable or “race” condition, oscillating between 00 and 11 undefined states
SR Latch behavior

1. start with $R = 0$, $S = 0$, the stored state is initially unknown
2. $S$ changes to 1, this sets $Q$ to 1
3. $S$ back to 0, $Q$ “remembers” 1 thus, two input conditions cause the system to be in set state
4. $R$ changes to 1, this resets $Q$ to 0
5. $R$ back to 0, now $Q$ “remembers” 0 thus, two input conditions cause the system to be in reset state
6. suppose both $S$ and $R$ changes to 1
7. both $Q$ and $\overline{Q}$ are zero!!!, undefined state
8. if both $R$ and $S$ go to zero simultaneously, can lead to unstable or “race” condition, oscillating between 00 and 11 undefined states

<table>
<thead>
<tr>
<th>Time</th>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>$\overline{Q}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
SR Latch behavior

1. start with $R = 0, S = 0$, the stored state is initially unknown
2. $S$ changes to 1, this sets $Q$ to 1
3. $S$ back to 0, $Q$ “remembers” 1 thus, two input conditions cause the system to be in set state
4. $R$ changes to 1, this resets $Q$ to 0
5. $R$ back to 0, now $Q$ “remembers” 0 thus, two input conditions cause the system to be in reset state
6. suppose both $S$ and $R$ changes to 1
7. both $Q$ and $\bar{Q}$ are zero !!!, undefined state
8. if both $R$ and $S$ go to zero simultaneously, can lead to unstable or “race” condition, oscillating between 00 and 11 undefined states
Race conditions:

1. $S = R = 1$, $Q = 0$, $\bar{Q} = 0$
2. $S$ and $R$ go simultaneously to 0
3. 1 gate delay later $Q = 1$, $\bar{Q} = 1$
4. 1 gate delay later $Q = 0$, $\bar{Q} = 0$
5. .....
Race conditions:

1. \( S = R = 1, \ Q = 0, \ \bar{Q} = 0 \)
2. \( S \) and \( R \) go simultaneously to 0
3. 1 gate delay later \( Q = 1, \ \bar{Q} = 1 \)
4. 1 gate delay later \( Q = 0, \ \bar{Q} = 0 \)
5. .....
Timing diagram

Race conditions:

1. $S = R = 1, \ Q = 0, \ \bar{Q} = 0$
2. $S$ and $R$ go simultaneously to 0
3. 1 gate delay later $Q = 1, \ \bar{Q} = 1$
4. 1 gate delay later $Q = 0, \ \bar{Q} = 0$
5. .....
Timing diagram

Race conditions:

1. $S = R = 1$, $Q = 0$, $\bar{Q} = 0$
2. $S$ and $R$ go simultaneously to 0
3. 1 gate delay later $Q = 1$, $\bar{Q} = 1$
4. 1 gate delay later $Q = 0$, $\bar{Q} = 0$
5. ..... 

In practice, it is very difficult to observe the SR Latch in the 1-1 state since one $S$ or $R$ usually changes first. The latch ambiguously returns to state 0-1 or 1-0.
The time behavior of the SR Latch is summarized in the state table showing next state based on the current inputs \((S, R)\) and the current state \(Q(t)\):

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>(Q(t))</th>
<th>(Q(t + \Delta))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
</tbody>
</table>

It can also be described by the following equation:

\[
Q(t + \Delta) = S + \bar{R}Q(t)
\]

\(\Delta\) is the gate delay, the time between change in input and corresponding change in state. One usually writes \(Q(t + 1)\).
Basic \(\overline{S\overline{R}}\) Latch

The cross-coupling of 2 NAND gates presents a similar behavior with:

- \(S = 0\) to switch to set state
- \(R = 0\) to switch to reset state
- both \(R = 0\), \(S = 0\) corresponds to an undefined state

\[ Q(t + 1) = \overline{S} + RQ(t) \]
Controlled SR Latch

- A control or ‘enable’ or ‘clock’ input \( C \) is added
- The state can only change if the control input is high
- The \( S, R \) inputs are only observed when \( C \) is high
- The behavior and the state table are exactly the same as those of the SR Latch (with NOR gates) when \( C = 1 \)
- When \( C = 0 \), the state remains unchanged, regardless of the values of \( S \) and \( R \)
- The problem of the undefined state remains: \( C = 1, S = 1, R = 1 \)

\[
\begin{align*}
S' &= S \\
R' &= R \\
S &= S' \\
R &= R'
\end{align*}
\]

The solution is: il faut une solution!
D Latch = Une Solution

- The undefined state is removed by imposing necessarily different values to inputs $S$ and $R$
- To this end, an inverter is added
- There remains one input $D$:
  - $D = 1$ is equivalent to $S = 1$
  - $D = 0$ is equivalent to $R = 1$
D latch: modes of operation

- When \( C = 0 \): the latch is in its **memorizing mode**, the output is the memorized state
- When \( C = 1 \): the latch is in its **transparent mode**, the output follows the input

\[
\begin{array}{ccc|c}
C & D & Q(t) & Q(t + 1) \\
\hline
0 & X & 0 & 0 & \text{memorizing mode} \\
0 & X & 1 & 1 \\
1 & 0 & 0 & 0 & \text{reset} \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & \text{set} \\
1 & 1 & 1 & 1 \\
\end{array}
\]

\[
Q(t + 1) = \overline{C}Q(t) + CD
\]
1. Fundamentals of Sequential Circuits
   1.1 Motivation
   1.2 Synchronous and Asynchronous Circuits
   1.3 State, State Diagram and State Table
   1.4 Time simulation

2. Latches

3. Flip-Flops

4. State diagrams and State Tables

5. Finite State Machine Diagrams
How to build a synchronous basic memory cell?

- Using latches, the states can continuously change, following their input changes as long as the clock signal is high.
- This undesired behavior for synchronous systems is linked to the feedback path from latches outputs to latches inputs through a combinational logic circuit.
- This feedback path imposes the delay of the combinational logic circuit which computes latches inputs from present state.

![Diagram of synchronous memory cell with feedback path and delay elements.](image)
The latch timing problem

The simplest possible combinational circuit is an inverter

The following simple circuit combines a D latch as memory cell and an inverter as combinational circuit

Suppose initially $Y = 0$

As long as $C = 1$, the value of $Y$ continues to change

The changes are based on the delay present in the loop through the connection from $Y$ to $Y$

Be careful

- the memory cell used is a stable circuit since it does not include an inverter in its internal feedback loop
- the problem comes from the external loop
Solution: the Flip-Flop

- The **desired behavior** for synchronous clocked sequential circuits: the state can only change **once** per clock pulse.
- The solution is to **break the closed path** from the input to the output of the storage element.
- Use Flip-Flop instead of latch.

Two types, depending on the triggering behavior:

- **Master-Slave Flip-Flop**: the state change is triggered by the value (high or low) of the clock.
- **Edge-Triggered Flip-Flop**: the state change is triggered by the positive (from 0 to 1) or negative (from 0 to 1) edge of the clock.
S-R Master-Slave Flip-Flop

The S-R Master-Slave Flip-Flop is made of two clocked SR latches connected in cascade. The clock is inverted for the second latch.

- **When \( C = 1 \):**
  - The first latch (master latch) is in its transparent mode.
  - The input is observed from the first latch and passed to output \( Y \).
  - The second latch (slave latch) is in its memorizing mode.
  - The past state is memorized, new state \( Y \) cannot pass to \( Q \).

- **When \( C = 0 \):**
  - The first latch is in its memorizing mode.
  - Any change in inputs \( S \) or \( R \) is not observed by \( Y \).
  - The second latch is in its transparent mode.
  - \( Y \) memorized by the first latch is passed to \( Q \).

In both cases, the path from inputs \( S \) and \( R \) to output \( Q \) is broken.

As in SR latch, \( S = R = 1 \) is not allowed.
SR Flip-Flop: timing behavior

<table>
<thead>
<tr>
<th>Clock</th>
<th>Master</th>
<th>Slave</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Memo</td>
<td>Transp</td>
</tr>
<tr>
<td></td>
<td>Transp</td>
<td>Memo</td>
</tr>
<tr>
<td></td>
<td>Memo</td>
<td>Transp</td>
</tr>
<tr>
<td></td>
<td>Transp</td>
<td>Memo</td>
</tr>
</tbody>
</table>

Transfer from master to slave

Duration of Memorization

PASSAGE DE L'INFO

C
S
R
Y
Q

\[ S = 1 \text{ AU FRONT DE } C \]
\[ R = \neg C \]

⇒ SET/RESET BIEN MARQUÉ
CAS IDÉAL DÉSIRÉ
The problem of “1s catching”

- The inputs $S$ and/or $R$ are allowed to change while $C = 1$

- Suppose $Q = 0$, if
  1. $S$ goes to 1 then back to 0 and then $R$ goes to 1 and back to 0 while $C$ still at 1
     - $Y$, output of the master latch, follows and goes to 1 and then to 0 after $R = 1$
     - finally 0 is passed to slave and $Q = 0$
  2. $S$ goes to 1 then back to 0 and then $R$ remains 0
     - $Y$ is set to 1 and does not change anymore
     - 1 is passed to slave and $Q = 1$

- The expected behavior is that $Q$ corresponds to the input values just before the clock goes to 0
- Case 1 is OK but unreliable
- Case 2 does not provide the expected output since $Q$ was zero before the clock pulse and $S$ and $R$ are both zero just before the clock goes to 0

$\Rightarrow ON \ VOUlAI'IT \ S=1 \Rightarrow Y??
Pulse-Triggered Flip-Flop vs. Edge-triggered Flip-Flop

- In SR Master-Slave flip-flop, any change in $S$, $R$ during $C = 1$ is taken into account: the 1s’ catching problem $+ S=R=1 \Rightarrow$ Non Defined
- The new state is triggered by the value of the clock
- This behavior is sometimes difficult to master: if
  - delays in combinational circuits are too high
  - or unintentional changes occur
- Unexpected state changes can be observed
- Better use Edge-Triggered flip-flops
- The state change is triggered by the transition of the clock
  - positive edge-triggered flip-flop: when $C$ goes from 0 to 1 (rising edge)
  - negative edge-triggered flip-flop: when $C$ goes from 1 to 0
Negative Edge-Triggered D Flip-Flop

The master clocked SR latch is replaced by a D latch

▶ When $C = 1$:
  ▶ The input is observed from the first latch and passed to output $Y$ as long as $C$ is high
  ▶ The SR latch latch is in its *memorizing* mode
  ▶ The past state is memorized, new state $Y$ cannot pass to $Q$

▶ When $C = 0$:
  ▶ The SR latch is in its *transparent* mode
  ▶ $Y$ memorized by the D latch at the time instant just before clock transition from 1 to 0 is passed to $Q$
  ▶ The change of the D flip-flop output is associated with value of input $D$ at the negative edge of the pulse

No 1s’ catching problem

CE QU’ON VEUT

GRÂCE AU D-LATCH!
Positive Edge-Triggered D Flip-Flop

- An inverter is added to the clock input
- \( Q \) changes to the value of \( D \) applied at the positive clock edge
Standard symbols for storage elements

(a) Latches

(b) Master-Slave Flip-Flops

(c) Edge-Triggered Flip-Flops
JK Flip-Flop

- The behavior of the JK flip-flop is analogous to the SR master-slave flip-flop except that $J = K = 1$ is allowed.
- For $J = K = 1$, the flip-flop changes to the complemented state.
- As a master slave flip-flop it has the same 1s’ catching behavior as the SR flip-flop.
- An edge-triggered JK flip-flop is preferred.
- It uses an edge-triggered D flip-flop.
T Flip-Flop

- The behavior is the following
  - for $T = 0$, no change in state $Q(t++) = Q(t)$
  - for $T = 1$, change to the complemented state $Q(t++) = \overline{Q(t)}$
- master-slave or edge-triggered
- the master-slave presents the 1s’ catching problem
- the edge-triggered is made from an edge-triggered D flip-flop
Asynchronous direct inputs

- At power up or when needed, all or part of a sequential circuit has to be initialized to a known state before it begins operation.
- This initialization is done **asynchronously**, independently of the clocked behavior.
- Direct $R$ and/or $S$ inputs are used.
- They control the state of the internal latches of the flip-flop.
- For example:

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>C</th>
<th>D</th>
<th>Q</th>
<th>$\overline{Q}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>XX</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>XX</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>XX</td>
<td>Undefined</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$\uparrow$ 0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$\uparrow$ 1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- 0 applied to $R$ input resets the flip-flop to the 0 state.
- 0 applied to $S$ input sets the flip-flop to the 1 state.
The following slides summarize the essential characteristics of the flip-flops, in terms of the:

- **Characteristic table**: this table defines the next state of the flip-flop in terms of the flip-flop inputs and current state.
- **Characteristic equation**: the equation that defines the next state of the flip-flop as a Boolean function of the flip-flop inputs and current state.
- **Excitation table**: this table defines the flip-flop input variables values needed to trigger a transition from the current state to the next state.
D Flip-Flop

- **Characteristic Table**
  
<table>
<thead>
<tr>
<th>D</th>
<th>Q(t+1)</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Reset</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Set</td>
</tr>
</tbody>
</table>

- **Characteristic Equation**
  
  \[ Q(t+1) = D \]

- **Excitation Table**
  
<table>
<thead>
<tr>
<th>Q(t+1)</th>
<th>D</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Reset</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Set</td>
</tr>
</tbody>
</table>
T Flip-Flop

- **Characteristic Table**
  
<table>
<thead>
<tr>
<th>T</th>
<th>Q(t+1)</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Q(t)</td>
<td>No change</td>
</tr>
<tr>
<td>1</td>
<td>\overline{Q}(t)</td>
<td>Complement</td>
</tr>
</tbody>
</table>

- **Characteristic Equation**
  
  \[ Q(t+1) = T \oplus Q \]

- **Excitation Table**

<table>
<thead>
<tr>
<th>Q(t+1)</th>
<th>T</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q(t)</td>
<td>0</td>
<td>No change</td>
</tr>
<tr>
<td>\overline{Q}(t)</td>
<td>1</td>
<td>Complement</td>
</tr>
</tbody>
</table>
SR Flip-Flop

- **Characteristic Table**

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q(t+1)</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Q(t)</td>
<td>No change</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Reset</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Set</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>?</td>
<td>Undefined</td>
</tr>
</tbody>
</table>

- **Characteristic Equation**

$$Q(t+1) = S + R Q, \quad + \quad S \cdot R = 0$$

- **Excitation Table**

<table>
<thead>
<tr>
<th>Q(t)</th>
<th>Q(t+1)</th>
<th>S</th>
<th>R</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>No change</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Set</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Reset</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>No change</td>
</tr>
</tbody>
</table>
JK Flip-Flop

- **Characteristic Table**

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
<th>Q(t+1)</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Q(t)</td>
<td>No change</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Reset</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Set</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Q(t)</td>
<td>Complement</td>
</tr>
</tbody>
</table>

- **Characteristic Equation**

\[
Q(t+1) = J \overline{Q} + \overline{K} Q
\]

- **Excitation Table**

<table>
<thead>
<tr>
<th>Q(t)</th>
<th>Q(t+1)</th>
<th>J</th>
<th>K</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>No change</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>Set</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>Reset</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>No Change</td>
</tr>
</tbody>
</table>
Flip-flop timing behavior : D, T
Flip-flop timing behavior: SR, JK
Fundamentals of Sequential Circuits
1.1 Motivation
1.2 Synchronous and Asynchronous Circuits
1.3 State, State Diagram and State Table
1.4 Time simulation

Latches

Flip-Flops

State diagrams and State Tables (ANALYSE)

Finite State Machine Diagrams
Modeling of a sequential circuit

The general model comprises a combinational circuit and a set of basic storage elements.

- **Synchronous** systems:
  - are driven by a *clock*
  - use flip-flops as storage elements
  - all the flip-flops must share exactly the same clock signal
  - all the flip-flops store their information at the same time

![Diagram with inputs, combinatorial circuit, outputs, flip-flops, clock, state, and next state connections]
Sequential systems are also called **Finite State Machines**

Two models depending on the way outputs are obtained:

- **Moore Model**
  - The outputs are a function **only of states**
  - \( \text{output}(t) = f(\text{state}(t)) \)

- **Mealy Model**
  - The outputs are a function of states **and inputs**
  - \( \text{output}(t) = f(\text{state}(t), \text{inputs}(t)) \)
State definition

- A state remembers meaningful properties of past input sequences that are essential to predict future output values.
- Example: state A represents the fact that a sequence of two successive “1” has occurred as the most recent past two inputs.
- Each of the states has been coded in binary values.
- To represent \( m \) states, we need \( n \) bits with \( n \geq \lceil \log_2 m \rceil \).
- Each bit corresponds to a state variable.
- A state is defined by a combination of values of the state variables.
- The state is linked to the flip-flops present in the circuit.
- One flip-flop contributes for one state variable.
- Example: the design of a sequential system requires 4 states.
  - The representation of these 4 states requires 2 bits.
  - 2 flip-flops and 2 state variables A and B.
  - Four states:
    - S0 : 00
    - S1 : 01
    - S2 : 10
    - S3 : 11
State table and State diagram

The behavior of the system can be defined by:

▶ a logic diagram made of flip-flops and usual combinational gates
▶ the combinational part of the circuit is characterized by flip-flop
  input equations: the Boolean functions that define the inputs of the
  flip-flops
▶ a State Table: similar to the truth table for combinational circuits.
  The state table presents
  ▶ the next state, i.e. the values of state variables at time \( t + 1 \)
  ▶ the values of the outputs at time \( t \)
  for all possible combinations of values of:
    ▶ the present state variables, at time \( t \) and
    ▶ the inputs at time \( t \)
▶ a State Diagram: a graphical form of the state table.
  ▶ Each state is represented by a circle, with the state name inside
  ▶ For each possible state transition, triggered by changes in the inputs,
    an arc is drawn from the present state to the next state
  ▶ Each arc is labeled with the inputs values that cause the state
    transition
  ▶ The corresponding outputs values are added to the labels or added
    to the state name (Moore model)
Example 1: Mealy model

Input: $X(t)$
Output: $Y(t)$
States: defined by $A(t)$ and $B(t)$, the two state variables

Four states:
- $S_0 : 00$
- $S_1 : 01$
- $S_2 : 10$
- $S_3 : 11$
Example 1: flip-flop input equations and output equation

- **flip-flop input equations:**
  \[
  A(t+1) = A(t)X(t) + B(t)X(t) \\
  B(t+1) = \overline{A(t)}X(t)
  \]

- **output equation:**
  \[
  Y(t) = \overline{X(t)}B(t) + \overline{X(t)}A(t)
  \]

- the D flip-flops used indicate that input, output and state are defined at positive edge of the clock.
State table

<table>
<thead>
<tr>
<th>Present State</th>
<th>Input</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A  B</td>
<td>X</td>
<td>A  B</td>
<td>Y</td>
</tr>
<tr>
<td>0  0</td>
<td>0</td>
<td>0  0</td>
<td>0</td>
</tr>
<tr>
<td>0  0</td>
<td>1</td>
<td>0  1</td>
<td>0</td>
</tr>
<tr>
<td>0  1</td>
<td>0</td>
<td>0  0</td>
<td>1</td>
</tr>
<tr>
<td>0  1</td>
<td>1</td>
<td>1  1</td>
<td>0</td>
</tr>
<tr>
<td>1  0</td>
<td>0</td>
<td>0  0</td>
<td>1</td>
</tr>
<tr>
<td>1  0</td>
<td>1</td>
<td>1  0</td>
<td>0</td>
</tr>
<tr>
<td>1  1</td>
<td>0</td>
<td>0  0</td>
<td>1</td>
</tr>
<tr>
<td>1  1</td>
<td>1</td>
<td>1  0</td>
<td>0</td>
</tr>
</tbody>
</table>

or in more compact form

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A  B</td>
<td>X = 0  X = 1</td>
<td>X = 0  X = 1</td>
</tr>
<tr>
<td>A  B</td>
<td>A  B</td>
<td>Y</td>
</tr>
<tr>
<td>0  0</td>
<td>0  0  0  1</td>
<td>0  0</td>
</tr>
<tr>
<td>0  1</td>
<td>0  0  1  1</td>
<td>1  0</td>
</tr>
<tr>
<td>1  0</td>
<td>0  0  1  0</td>
<td>1  1</td>
</tr>
<tr>
<td>1  1</td>
<td>0  0  1  0</td>
<td>1  0</td>
</tr>
</tbody>
</table>
State Diagram

- 4 states S0, S1, S2, S3
- one circle for each state
- 2 transitions from each state corresponding to $X = 0$ and $X = 1$

The diagram, and thereof the problem, can be further simplified by merging equivalent states.

L'EST-CE?
Equivalent state definitions

- Two states are equivalent if their response for each possible input sequence is an identical output sequence
- Or equivalently, two states are equivalent if their outputs produced for each input value is identical and their next states for each input value are the same or equivalent

- States S2 and S3 are equivalent, same output and identical next state for $X = 0$ and $X = 1$
- S2 and S3 are merged in a new state $S'2$
- The new state $S'2$ and S1 are also equivalent
- They are merged into new state $S'1$
- Finally, 2 states remain which can be implemented using only one bit, and thus one state variable
- The system can be redesigned using only one flip-flop
Example 1: Moore and Mealy models

- The system combines Mealy and Moore models
  - in state S0, the output does not depend on the input (always 0): Moore model
  - the output value is removed from the label on the arc and included in S0 circle
  - for the other states, the output depends on the input: Mealy model
Example 2 - Moore model

- The system is defined by its flip-flop input equation and its output equation
- inputs: $X$ and $Y$
- output: $Z$
- state variable: $A$

\[
A(t + 1) = A(t) \oplus X(t) \oplus Y(t)
\]
\[
Z(t) = A(t)
\]
Fundamentals of Sequential Circuits

1.1 Motivation
1.2 Synchronous and Asynchronous Circuits
1.3 State, State Diagram and State Table
1.4 Time simulation

2. Latches
3. Flip-Flops

4. State diagrams and State Tables

5. Finite State Machine Diagrams

- STATE MACRO
- STATE THRUST (z = y)
- STATE V.R. (z = y)
- LOGIC DIAGRAM (F.F.)

- SEA SYSTEM
- @ of
- LEVEL VS EDGE TRIGGERED
- (S, A, C, S, A) (G, C, G, ...)
- ALERT VS FF

- SUM UP
Références

▶ *Logic and Computer Design Fundamentals, 4/E*, M. Morris Mano Charles Kime, Course material
http://writphotec.com/mano4/

▶ *Cours d’électronique numérique*, Aurélie Gensbittel, Bertrand Granado, Université Pierre et Marie Curie
http://bertrand.granado.free.fr/Licence/ue201/coursbeameranime.pdf

▶ *Lecture notes, Course CSE370 - Introduction to Digital Design*,
Spring 2006, University of Washington,
https://courses.cs.washington.edu/courses/cse370/06sp/pdfs/
Terms of Use

- All (or portions) of this material © 2008 by Pearson Education, Inc.
- Permission is given to incorporate this material or adaptations thereof into classroom presentations and handouts to instructors in courses adopting the latest edition of Logic and Computer Design Fundamentals as the course textbook.
- These materials or adaptations thereof are not to be sold or otherwise offered for consideration.
- This Terms of Use slide or page is to be included within the original materials or any adaptations thereof.