Feedback and Flip-Flops

Philipp Koehn

7 September 2019
The Story So Far

• We can encode numbers

• We can do calculation

• ... but it’s all a bit static

• How about a counter?

→ this requires "memory"
feedback
A Strange Contraption
Let’s Turn It On

Electricity is on → this opens the normally closed key
Let’s Turn It On

Electricity is off → this closes the normally closed key
What Do We Have?

• A Buzzer

• A Clock

• An Oscillator

(symbol)
Oscillator

- **Period** of oscillator

- Frequency: cycles per second

- Unit: 1 cycle per second: 1 Hertz

- Modern computes:
  Billions of Hertz = Gigahertz (GHz)

Heinrich Hertz
1857--1894
flip flop
Another Contraption
Closing Upper Key

V
NOR
V
NOR
OUT
Opening Upper Key

Same key configuration as initially

**But:** Now OUT is on --- we *remembered* the key turn
Closing Lower Key
Opening Lower Key

Back to initial state
• We have memory -- called **Reset-Set Flip-Flop**

• Truth table

<table>
<thead>
<tr>
<th>UPPER</th>
<th>LOWER</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>OUT</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Illegal</td>
</tr>
</tbody>
</table>

• **UPPER** = **SET**

• **LOWER** = **RESET**
Re-Arranged
Symmetric

\[ S \rightarrow \text{NOR} \rightarrow \overline{Q} \]

\[ R \rightarrow \text{NOR} \rightarrow Q \]
<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>\bar{Q}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Q</td>
<td>\bar{Q}</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Illegal</td>
<td></td>
</tr>
</tbody>
</table>
d-type flip flop
Vision

- Control bit ("clock")
  - on = write to memory
  - off = read from memory

- Data bit
  - data item to be written

- Output
  - current state of the memory
Replace Set/Reset with Data

\[\text{DATA} \xrightarrow{\text{NOR}} Q \xrightarrow{\text{NOR}} \overline{Q} \]
Add Control Bit ("Clock")

DATA

CLOCK

AND

NOR

AND

NOR

Q

Q

Q

Q
D-Type Flip-Flop

- Also called **D-type latch**

- Circuit latches on one bit of memory and keeps it around

- Truth table

<table>
<thead>
<tr>
<th>Data</th>
<th>Clock</th>
<th>Q</th>
<th>(\bar{Q})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>Q</td>
<td>(\bar{Q})</td>
</tr>
</tbody>
</table>

- Can also build these for multiple data bits
accumulative adder
Design Goal

- Adder has initially value 0

- Adding a number
  -> value increases

- Resetting
  -> value goes back to 0
Ingredients

8-BIT ADDER

A7 A6 A5 A4 A3 A2 A1 A0 B7 B6 B5 B4 B3 B2 B1 B0
S7 S6 S5 S4 S3 S2 S1 S0
CO CI

8-BIT LATCH

D7 D6 D5 D4 D3 D2 D1 D0
Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0
CLK
Ingredients

8-BIT ADDER

\[
\begin{array}{c}
A \\
B \\
S \\
C_{O} \\
C_{I}
\end{array}
\]

8-BIT LATCH

\[
\begin{array}{c}
D \\
Q \\
C_{L K}
\end{array}
\]
Building an Accumulative Adder

• Latch: current sum
• Clock on → set it to 0
Building an Accumulative Adder

- Adder
- Combines
  - current value
  - selected input
Building an Accumulative Adder

- Can we pass output directly to latch?

- Concerns
  - select between 0 and sum
  - only stored when clock on
Building an Accumulative Adder

- 2-1 selector
- Either uses 0 or sum
- Built with AND gates
- Still have runaway feedback loop...
Building an Accumulative Adder

- Two Latches
  - one to store the sum
  - one to store input to adder

- Clock 1
  - carry out addition
  - store result

- Clock 2
  - transfer to set up next addition
Building an Accumulative Adder

- Combine the clocks
- Pressing the add key
  - carry out addition
  - store result in upper latch
- Release the add key
  - transfer to lower latch
  - set up next addition
• Remember the oscillator?

\[ \text{NOT} \]

\[ \text{IN} \rightarrow \text{OUT} \]
• Each cycle of oscillator:
  keeps adding
What Else?

- We have something interesting here
edge triggered flip-flop
D-Type Latch

- When clock is on, save data
- "Level-triggered"
• "Edge-triggered": changes value, when switched from 0 to 1
Edge Triggered D-Type Latch

Symbol
## Truth Table

<table>
<thead>
<tr>
<th>Data</th>
<th>Clock</th>
<th>Q</th>
<th>(\bar{Q})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>↑</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>↑</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>Q</td>
<td>(\bar{Q})</td>
</tr>
</tbody>
</table>
ripple counter
Oscillator and Latch

\[
\begin{array}{c|c|c|c}
\text{Data} & \text{Clock} & Q & \bar{Q} \\
1 & 0 & 0 & 1 \\
1 & \uparrow & 1 & 0 \\
0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 \\
0 & \uparrow & 0 & 1 \\
1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 \\
\end{array}
\]
Oscillator and Latch

<table>
<thead>
<tr>
<th>Data</th>
<th>Clock</th>
<th>Q</th>
<th>(\bar{Q})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>↑</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>↑</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
### Halving of Frequency

<table>
<thead>
<tr>
<th>Data</th>
<th>Clock</th>
<th>Q</th>
<th>(\bar{Q})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>↑</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>↑</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**IN**
```
IN
```

**OUT**
```
OUT
```
Multiple Bits

OUT0

OUT1

OUT2

OUT3

D ▶ Q

D ▶ Q

D ▶ Q

NOT

OUT0 OUT1 OUT2 OUT3

OUT0

OUT1

OUT2

OUT3
### Ripple Counter

<table>
<thead>
<tr>
<th>OUT0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUT1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>OUT2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>OUT3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</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>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |