9 Sequential Circuits

- In **combinational circuits** the output signals depend on current values of input signals.
- Past values of input signals do not have any influence on the current values of the output signals.
- In this sense, processing of input signals in combinational circuit is performed in a single time step.
- Combinational circuits are **static systems**.
- In **sequential circuits** input signals are processed in a multi-step way, so that output signals depend not only on current, but also on the past values of the input signals.
- This time dependency is implemented using the concept of internal **state signals**.
- Sequential circuits are dynamic systems, and their formal description is given by a model of a finite state machine.
- Any sequential circuit can be ultimately reduced to a single finite state machine. Therefore, these two terms are formally synonymous.

Sequential circuits are hierarchically classified into the following groups with respect to their growing structural complexity:

- Latches and flip-flops,
- Registers and counters,
- Algorithmic State Machines,
- Processors, consisting of a datapath and a control unit.

9.1 Finite State Machines

A finite state machine can be always reduced to

- a combinational logic block,
- feedback paths, possibly with delay elements in them

The combinational block has \( p + n \) inputs and \( m + n \) outputs.

The input, state and output signals are related by the following well known equations:

\[
Q(t + dt) = f(Q(t), X(t)) \quad \text{state equation (9.1)}
\]
\[
Y(t) = g(Q(t), X(t)) \quad \text{output equation (9.2)}
\]

- The **state equation** says that the “next” values of the state signals \( Q(t + dt) \) are a function \( f \) of the current values of state signals \( Q(t) \) and input signals \( X(t) \).
- The value of \( dt \) describes the delay in the feedback loop and can be controlled by an additional delay element \( D \).
- Every **combination of state signals** is called the **state** of the state machine.

With \( n \) state signal the maximum number of different states is \( 2^n \).
• The output equation says that the current values of output signals \( Y(t) \) are a function \( g \) of the current values of state signals \( Q(t) \) and input signals \( X(t) \).

• Note that the state transition function, \( f \), and the output function, \( g \), are combinational functions and “sequencing” is the result of the feedback loops.

• Functions \( f \) and \( g \) are often described in a tabular form. Such tables are called the state transition table, and output table, respectively.

• If the output function in eqn. (9.2) is of the form

\[
Y(t) = g(Q(t))
\]

then the state machine is called a Moore state machine.

• In simple cases, the output signals, \( Y \), can be just a subset of the state signals, \( Q \).

• Note that in a Moore state machine output signals are independent of input signals.

• In a general case as in eqn. (9.2), when output signals depend both on input and state signals, the machine is called a Mealy state machine.

• It can be shown that from the purely formal point of view both versions are equivalent and can be converted to each other.

• From the point of view of the way the states evolve with time, the state machines are classified into two groups named asynchronous and synchronous.

9.1.1 Asynchronous state machines/sequential circuits

can be characterized by the following properties

• there are no delay elements in the state signals feedback loops, and, as a result, the delay \( dt \) in eqn (9.1) is “irregular” and depends on the internal delays of the components forming the combinational block,

• after a change of a state or input signal, the change of the states (state signals) occurs instantaneously, after the internal delay \( dt \) of the combinational block,

• because of this property, it is difficult to design a stable asynchronous state machine.

• Typically, asynchronous state machines used to build simple sequential circuits like latches and flip-flops.

A necessary condition for an asynchronous state machine to be stable, is that there must exist combinations of input and state signals such that the next state \( Q(t + dt) \) is identical with the current state \( Q(t) \).

A necessary condition for an asynchronous state machine to move correctly between states is that during any transition only one state signal may be changed. Otherwise we have a race between state signals and the final state will be unpredictable.

• For brevity, the current state \( Q(t) \) and the next state \( Q(t + dt) \) are often denoted \( Q \) and \( Q^+ \).
9.1.2 Synchronous (clocked) state machines

- are characterized by a fixed delay in the feedback path and, as a result, the change of states occurs only on the rising (falling) edge of the clock signal.

In other words, the clock frequency, \( f_c \), specifies the delay \( dt = \frac{1}{f_c} = t_c \)

The state equation is written in the following form

\[ Q(t+1) = f(Q(t), X(t)) \]  \hspace{1cm} (9.4)

where \( t \) is now an integer numbering the clock period instances.

- We will see that the delay block in the general state machine is replaced with a state registers \( Q \) triggered by a specialised input signal, the clock signal, \( clk \).

- A combinational circuit converts \( p \) input signals and \( n \) state signals into \( m \) next state signals.

- The register \( Q \) holds the state signals steady between the clock edges, that is for one clock cycle.

- A register in itself is a simple sequential circuit.

\[ Q(t+1) = f(Q(t), X(t)) \]  \hspace{1cm} (9.4)

9.2 Latches

Latches are the simplest bi-stable (two states) asynchronous sequential circuits and are structurally characterized by the presence of a single feedback loop for the single state signal, \( Q \).

The output signal is identical with the state signal, hence latches are Moore state machines.

9.2.1 S-R Latch

The S-R latch has two inputs: \( S \) – set and \( R \) – reset, and one state signal \( Q \) which is also the output signal.

Operations of the latch are best described by its state (transition) diagram which shows how the states evolves for different values of the input signals.

The state diagram clearly shows that there are two stable states, \( Q = 0 \) and \( Q = 1 \).
• The state diagram can now be converted into the **state table**.

• The table shows what is the **next state** $Q^+$ for the **current state** $Q$ and all combinations of **input signals**.

• Note that the signal $S = 1$ moves the latch to the $Q = 1$ state, whereas the signal $R = 1$ moves the latch to the $Q = 0$ state.

• Situation when both input signals are high, $S \cdot R = 1$, results in don’t care conditions.

\[
\begin{array}{|c|c|c|}
\hline
S & R & Q^+ \\
\hline
0 & 0 & 0 \\
0 & 1 & 1 \\
0 & 1 & 0 \\
1 & 0 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
1 & 1 & * \\
\hline
\end{array}
\]

• From the **state table** we can obtain the **state equations** $Q^+ = f(Q, S, R)$ using any minimization method, e.g. a Karnaugh map and subsequently the logic structure of the latch.

• Note the cross-coupled pair of NOR gates.

Finally, the S-R latch can be described through the time waveforms:

**Exercise:**

For a dual cross-coupled pair of NAND gates derive the state equations, state table and state diagram.
### 9.2.2 D Latch

- The D latch has **two inputs**: $D$ – data and $Ld$ – load, and one **state signal** $Q$ which is also the output signal.

- The operations if the **D latch** can be best described by the **state diagram**:
  - When $Ld = 0$, the latch maintains its current states.
  - When $Ld = 1$, the latch moves to the state: $Q = D$, that is, it stores data $D$ in its internal state $Q$.

- From the state diagram we can obtain by the following **state tables**, which give the values of the **next state signal**, $Q^*$, as a function of the **current state signal**, $Q$, and input signals $Ld$, $Q$:

  $$
  \begin{array}{|c|c|c|c|}
  \hline
  Ld & D & Q & Q^* \\ 
  \hline
  0 & 0 & 0 & 0 \\ 
  0 & 0 & 1 & 1 \\ 
  0 & 1 & 0 & 0 \\ 
  0 & 1 & 1 & 1 \\ 
  1 & 0 & 0 & 0 \\ 
  1 & 0 & 1 & 0 \\ 
  1 & 1 & 0 & 1 \\ 
  1 & 1 & 1 & 1 \\ 
  \hline
  \end{array}
  $$

- As previously, the state table can be converted in the state equations and subsequently into the logic structure of the latch.

---

**A possible implementation:**

- **Waveforms:**

---

A.P. Papliński
9.2.3 **Latches — Summary**

State Transition Diagrams for the SR and D latches:

![State Transition Diagrams](image)

C/L stands for “combinational logic”

- Note that a latch stores 1-bit data.

Possible logic diagrams of the S-R and D latches:

![Logic Diagrams](image)

The latches are referred to as the **level-sensitive** devices:

when the Ld signal is high, every change of the input D is immediately stored in (transfer to) the latch output, Q.
9.2.4 VHDL description

- In order to write a VHDL specification of a latch we can use its state table or any equivalent logic equation.

- For example, a possible description of the S-R latch using its “next-state” equation can have the following form:

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY LchSR IS
    PORT (S, R : IN std_logic;
           Q : OUT std_logic);
END Lchsr;

ARCHITECTURE nors OF Lchsr IS
    SIGNAL Qi : std_logic;
    BEGIN
        Qi <= not ( R or not( S or Qi) ) ;
        Q <= Qi;
    END nors;
```

- Note that since the port signal \( Q \) is an output signal it cannot be used in expressions.

- Therefore, we have to introduce an internal signal \( Qi \) identical in behaviour to \( Q \).

9.2.5 Simulation waveforms

- Simulation waveforms for an S-R latch

- As expected signal \( R \) resets the state of the latch \((Q=0)\) and \( S \) sets the state to \((Q=1)\).

- When both \( R \) and \( S \) are HIGH, the latch goes to \((Q=0)\).

- Simulation waveforms for the D latch:
9.3 Flip-Flops

- **Latches**, being level-sensitive devices, cannot be easily used to form more complex sequential circuits.
- Consider, for example a circuit in which the complement of the output of the D latch is fed back as the input signal.
- The resulting behaviour is oscillatory for $L_d = 1$, because of the loop that goes through three inverting gates.

- **Flip-Flops are edge-sensitive devises.**
- The change of the flip-flop state can occur only as the result of the (positive/rising) edge of the clock signal that synchronizes the work of the flip-flop.

9.4 D Flip-Flop

- The D Flip-Flop (D FF) is the most fundamental sequential circuit.
- It is the building block of registers, counters and synchronous state machines.

<table>
<thead>
<tr>
<th>D</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk</td>
<td>operation</td>
</tr>
<tr>
<td>↑</td>
<td>Q &lt;= D</td>
</tr>
<tr>
<td>others</td>
<td>Q &lt;= Q</td>
</tr>
</tbody>
</table>

- The D flip-flop is similar to the D latch in that the input signal, $D$, is loaded into the FF ($Q <= D$) under the control of the triggering signal ($L_d$, or $clk$).
- Unlike in the latch, the load operation in flip-flops takes place only during the rising edge of the clock signal.
- Outside the triggering edge of the clock signal the flip-flop is insensitive to any changes at its D input as indicated in the timing diagram.
- The waveforms illustrate the fact that the value of the input signal $D$ at the rising edge of the clock, say, $s_1$, is loaded into the flip-flop in that sense that $Q = s_1$.
- Changes in $D$ between the rising edges does not influence the state of the flip-flop.
• The state (transition) diagrams for synchronous circuits assume that any change of the state occurs only for the rising edge of the clock signal.

• Therefore, the clock signal is not shown in the state diagrams.

• Hence, the state diagram for the D Flip-Flop as a synchronous state machine is very simple:

Note some variation regarding description of the next-state conditions.

We write for example \( D = 0 \) or just \( \bar{D} \) whichever is more appealing to you.

9.4.1 VHDL description of a D flip-flop

• The VHDL description of flip-flops is relatively simple but a few new language structures need to be used.

Consider the following template describing the D flip-flop:

```vhdl
SIGNAL clk, D, Q : bit :
...
PROCESS (clk)
BEGIN
  IF (clk'EVENT AND clk = '1') THEN
    Q <= D;
  END IF;
END PROCESS;
```

• The process statement encapsulates sequential statements.

• The sensitivity list specify signals that are monitored for any change or modification.

• In VHDL the edge-detection mechanism is provided by the event attribute.

• This attribute operates on a signal and returns a boolean value which is always false, unless the signal showed a change in value, that is a signal edge.

• \( (clk'EVENT AND clk = '1') \) makes sure that it is the rising edge which is detected.

• Note also that only clk signal is on the process sensitivity list.

• The if . . then statement is a sequential statement.

• The process statement with the if statement makes sure that the assignment \( Q <= D \) takes place after the rising edge of the clock is detected.
9.4.2 Logic synthesis of a D flip-flop

- The positive-edge-triggered D flip-flop (D-FF) is the simplest, but the most important synchronous sequential circuit, the building block of registers, counters and synchronous state machines.

- Internally, however, such a flip-flop is an asynchronous sequential circuit (state machine).

- The design procedure is similar to that outlined for the D latch.

- The first parameter to establish is the number of states, or state signals needed to describe the sensitivity to the clock edge.

- Although formal methods to find out the required number of states do exist, we will just try first if two state signals, that is, four states are sufficient.

- Solutions with three state signals also exist.

A block-diagram, the state diagram and the symbol of a positive-edge-triggered D flip-flop:

- There are four states coded by two state signals, $P$ and $Q$.
- Two state signals are fed back to the inputs of the combinational circuit.
- There are two input signals, $D$ (data) and $CLK$ (clock). The $Q$ state signal is used as an output signal.

Note that the state transition diagram satisfies the two necessary conditions for a “sensible” asynchronous state machine, namely, that

- there are combinations of input and state signals such that the next state is identical with the current state, and that
- during transitions between states only one state signal is being changed.
The expected behaviour described by the state diagram is that during the rising edge of the clock signal the 1-bit data from the input \( D \) is loaded into the flip-flop, that is,

\[
Q \leq D
\]

Outside the rising edge of the clock signal the state \( Q \) is to be unchanged, regardless of the variations of the \( D \) input and possibly the second state signal, \( P \).

Assume that we are in the state \( PQ = 00 \).

If \( clk = 0 \), we remain in this state.

If \( clk = 1 \), we go either to state \( PQ = 01 \), or \( PQ = 10 \) depending on the value of the signal \( D \).

If \( clk = 1 \) and we have just move to the state \( PQ = 01 \) or \( 10 \), we stay in such a state regardless of the value of the signal \( D \). This describes the positive-edge-triggered behaviour.

In states \( PQ = 01 \) or \( 10 \) we wait for clock signal to go to zero and then we go either to the state \( PQ = 11 \) or \( 00 \), respectively.

The next step is to design a combinatorial circuit which implements the required behaviour of the flip-flop.

For this, we convert the state diagram into the state transition table. The state transition table describes the relationship between the current state signal, \( P, Q \), the input signals, \( clk, D \), and the next state signals \( P+, Q+ \). The symbol ‘–’ denotes the don’t care conditions.
• The state transition table is equivalent to the truth table for the following combinatorial circuits:

\[
P^+ = f_P(\text{clk}, D, P, Q) \\
Q^+ = f_Q(\text{clk}, D, P, Q)
\]

• If we aim at “manual” design, then the functions \(f_P\) and \(f_Q\) can be obtained from the following Karnaugh maps generated from the state transition table.

\[
\begin{array}{c|c|c|c|c}
\text{clk} & \text{D} & \text{P} & \text{Q} & \text{P}^+ \\
\hline
0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 \\
\end{array}
\hspace{1cm}
\begin{array}{c|c|c|c|c}
\text{clk} & \text{D} & \text{P} & \text{Q} & \text{Q}^+ \\
\hline
0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 \\
\end{array}
\]

• From the Karnaugh maps it is possible to derive a suitable gate implementation of the combinatorial part of the flip-flop.

• Two feedback loops, for \(P = P^+\) and \(Q = Q^+\), complete the design.

If we decide that it would be more efficient to rely on the VHDL compiler to synthesize a good combinational circuit, then a relevant VHDL code could be of the following form:

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

-- D flip-flop described by the truth table
ENTITY Dfftt IS
  PORT( clk, D : IN std_logic ;
        Q : OUT std_logic );
END Dfftt;
ARCHITECTURE ttbl OF Dfftt IS
  TYPE arr2d IS ARRAY (natural range <>,
                      natural range <>) OF std_logic;
  CONSTANT ttDff : arr2d(1 to 2, 0 to 15) := (
    -- truth table for P^+ Q^+
    --0123456789abcdef (P, Q, clk, D)
    "0010110000111110" , --1 P
    "0001111100001101" ); --2 Q
  SIGNAL P, QQ : std_logic ;
  SIGNAL S : std_logic_vector (3 downto 0) ;
  SIGNAL Si : integer range 0 to 15 ;
BEGIN
  S <= (P, QQ, clk, D) ;
  Si <= conv_integer(unsigned(S)) ;
  -- reading from the truth table
  P <= ttDff (1, Si) ;
  QQ <= ttDff (2, Si) ;
  Q <= QQ ;
END ttbl;
```

A.P. Papliński 9–23
9.4.3 Yet another implementation of a D flip-flop

- Another efficient implementation of a D flip-flop contains three state signals, \( S, R, Q \) as shown in the logic diagram.
- Three feedback loops are arranged so that the flip-flop consists of three interconnected S-R latches.
- Two “input” latches generates signals \( S \) and \( R \), respectively, from the \( CLK \) and \( D \) input signals.
- The third, “output”, low-level active latch is driven by signals \( S \) and \( R \).

Exercise

Consider an asynchronous sequential circuit with two inputs \( CLK \) and \( D \), and two state signals \( S \) and \( R \) as in the logic diagram above.

- Derive the state diagram and the state table.
  Note that with two state signals there are four possible states.

9.5 The T flip-flop

- The T (toggle) flip-flop is the second most important one after the D flip-flop.
- The D flip-flop is the building block of the registers.
- The T flip-flop is the building block of the counters.
- The symbol, operation table and the state diagram of a rising edge triggered T flip flop are as shown:
  - Remember that all state transitions/operations are performed at the rising edge of the clock signal.
  - The principle of operation is simple:
    - when the toggle signal is asserted \((T = 1)\) the rising edge of the clock signal changes the state of the flip-flop (the value of the state signal \( Q \)) to the opposite value.
    - Outside the rising edge, or when \((T = 0)\) the state remains unchanged.
9.5.1 Logic synthesis of the T flip-flop

- In order to design the logic structure of the T flip-flop we start with its state diagram:
- Comparing with the D flip-flop the state diagram of the T flip-flop is slightly simpler which should result in a simpler implementation.
- In order to design the flip-flop we convert the state diagram into the state transition table, possible in the form of a Karnaugh map.
- The resulting map is as follows:
- From the above table we can derive the following logic equations:
  \[ P^+ = Q \cdot \overline{Clk} + P \cdot Clk \]
  \[ Q^+ = Q \cdot \overline{Clk} + Clk \cdot (T \cdot \overline{P} + \overline{T} \cdot Q) \]
- The signal \( P^+ \) can be generated using a 2-to-1 multiplexer driven by \( Clk \) which switches between \( Q \) and \( P \).

9.5.2 Implementing the T flip-flop with a D flip-flop

- A T flip-flop can be built as a sequential circuit using a D flip-flop and a logic excitation circuit, as in the block-diagram.
- The excitation table is obtained from the state table and describes the value of the D flip-flip input signal \( D \) as a function of the state signal \( Q \) and the input signal \( T \).
  \[ D = f(Q, T) \]
  \[ D^+ = Q^+ = T \oplus Q \]
  \[ D = T \oplus Q \]
9.5.3 The T flip-flop as a frequency divider

One of the application of the T flip-flop is as a frequency divider.

Consider the following waveforms for $T = 1$:

Note that if the clock frequency is $f_c$ and the frequency of the output signal is $f_Q$, then we clearly have

$$f_Q = \frac{1}{2} f_c$$

9.6 The JK flip-flop

- The JK flip-flop was popular some time ago and combines the functions of a SR and T flip-flops.

- In addition to the clock signal it has two inputs $J$ (set) and $K$ (reset) and its operations can be described by the following table:

- Equivalently, the JK flip-flop can be described by the following (synchronous) state diagram in which transitions take place during the rising edge of the clock signal:

- The state diagram can be converted into the following (synchronous) state table:
- All synchronous (aka clocked) sequential circuits can be build using flip-flop(s), D flip-flop in particular, and a suitable excitation circuit which calculates the next state from the current state and input signals:

- If a D flip-flop is used, then the excitation table is identical to the state table, D replacing \( Q^{\pm} \):

- The resulting excitation equation is as follows:

\[
D = J \cdot \bar{Q} + \bar{K} \cdot Q = (J + Q) \cdot (K \cdot Q) = (J + Q) + K \cdot Q
\]

- A possible logic diagram follows:

9.7 Hazards in combinational and sequential circuits

as presented in sec. 9-7 of the textbook.