8 Sequential logic circuits – bistable elements

8.1 Objectives

The synthesis and functionality of common bistable elements is studied. For the synthesis part, the implementation is strictly NAND-based. The functionality highlights their specific behavior according to input commands, which can be asynchronous or synchronous. The advantages and disadvantages concerning different types of clock signal synchronization (e.g. level based, or edge based) are detailed. Finally, a method for implementing a bistable with other types of bistables, and logic gates, is presented.

8.2 Theoretical considerations

The sequential logic circuits are level 1 automata characterized by internal state and outputs, which can be controlled using input commands. The simplest sequential circuits are the bistable elements, also known as *bistables*. They have two distinct states coded as 0 and 1, meaning they can store one bit of data. When powered, the state of the circuit can be maintained indefinitely or can be changed. The bistables have two complementary outputs, representing the internal state and its inverted value, respectively.

Considering the reaction from the input commands the bistables are classified as a*synchronous* and *synchronous*. The asynchronous bistables have an immediate reaction, when triggered. The synchronous bistables react depending on the *clock signal* state. The clock is a signal oscillating between 0 and 1, with a period T and a frequency *f*=1/T. The synchronous bistables can have asynchronous inputs with immediate effect on their state, when activated. These inputs have priority over synchronous commands. Usually, the asynchronous inputs are used to initialize the state of the bistable to 0 or 1.

8.2.1 The asynchronous SR latch

The asynchronous SR latch has two asynchronous inputs, one for set (S) and another for reset (R). The function table is highlighted in Table 8. 1.

Table 8. 1 Function table of the asynchronous SR latch (Q^t = current state, Q^{t+1} = next state)

S	R	Q^{t+1}
0	0	Q^t
0	1	0
1	0	1
1	1	*

The synthesis with NAND gates can be obtained from the Minimal Disjunctive Normal Form (MDNF) by inverting twice and applying De Morgan law (Figure 8. 1). **Note**: Inputs Sn and Rn are active low.

Q^t	S	R	Q^{t+1}	$\overline{Q^{t+1}}$
0	0	0	0	1
0	0	1	0	1
0	1	0	1	0
0	1	1	X	X
1	0	0	1	0
1	0	1	0	1
1	1	0	1	0
1	1	1	X	X

inverted twice + De Morgan

$$Q^{t+1} \xRightarrow{\text{MDNF}} S + (Q^t \cdot \overline{R}) = \overline{\overline{S} \cdot \overline{(Q^t \cdot \overline{R})}}$$

$$\overline{Q^{t+1}} \xRightarrow{\text{MDNF}} R + (\overline{Q^t} \cdot \overline{S}) = \overline{\overline{R} \cdot \overline{(\overline{Q^t} \cdot \overline{S})}}$$

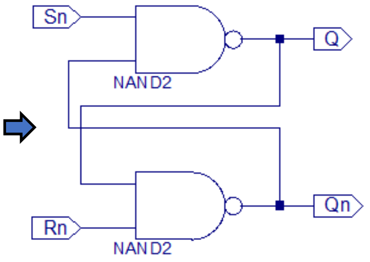
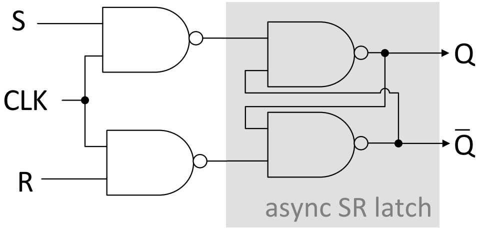


Figure 8. 1 Synthesis of the asynchronous SR latch using NAND gates (Sn=$\overline{S}$, Rn=$\overline{R}$, Qn=$\overline{Q}$)

8.2.2 The synchronous SR gated-latch

The SR gated-latch has a clock signal CLK, and two active high inputs S and R, implementing the functionality in Table 8. 1. Signals R and S are synchronized with the clock signal, meaning they can only affect the bistable state when CLK=1. This type of circuit can be implemented using NAND gates, by extending the implementation of the asynchronous SR latch according to Figure 8. 2.



Q^t	Q^{t+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Figure 8. 2 Synthesis of the synchronous SR gated-latch using NAND gates (left) and its excitation table (right)

The *excitation table* in Figure 8. 2 can be extracted from Table 8. 1. It highlights the values on the input commands S, R, necessary for a certain behavior on the bistable state. These values are defined for all combinations of current and future states. For instance, the top row highlights that keeping the state to 0 (Q^t=0, Q^{t+1}=0) is possible when S=0 and R=X (R can be 0 or 1). For the second row, changing the state from 0 to 1 (Q^t=0, Q^{t+1}=1), requires S=0 and R=1 on inputs.

8.2.3 The D gated-latch and D flip-flop (D ≈ data/delay)

The D gated-latch has one synchronous input called D. When CLK=1 the value on D is transferred to the state of the bistable. The symbol of the circuit and its synthesis with NAND gates are presented in Figure 8. 3.

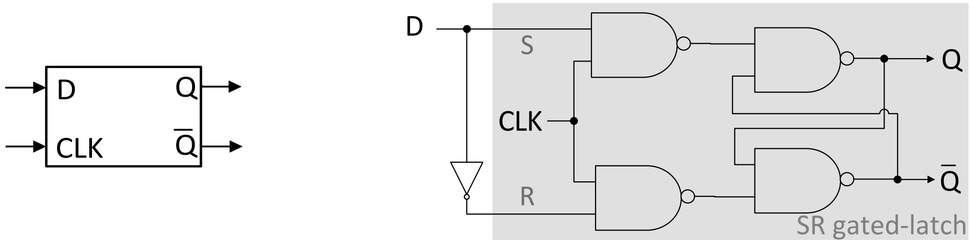


Figure 8. 3 The symbol of the D gated-latch (left) and its NAND-based synthesis (right)

Note: A D-latch can be implemented using an SR bistable with S=D and R=$\overline{D}$.

Both the function table and the excitation table (Table 8. 2) expose the expression $Q^{t+1} = D$, which describe the future state as a function of the input D. The expression is also called the *characteristic equation* of the D bistable.

Table 8. 2 The function table (left) and the excitation table (right) of the D bistable

CLK	D	Q^{t+1}
0	X	Q^t
1	0	0
1	1	1

Q^t	Q^{t+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

The behavior of the bistable is highlighted in the next timing diagram. The input D affects the bistable state when CLK=1 – the trigger is on the *positive level* – and it is blocked when CLK=0. Both intervals are marked with green and red in the diagram.

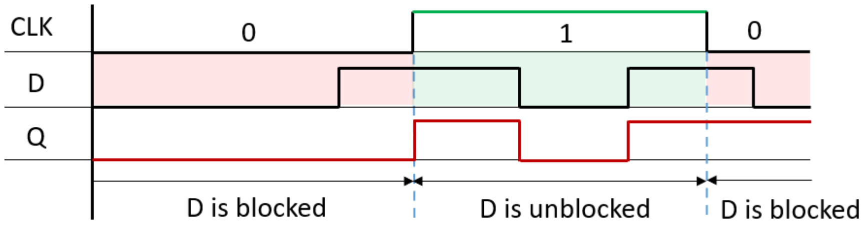
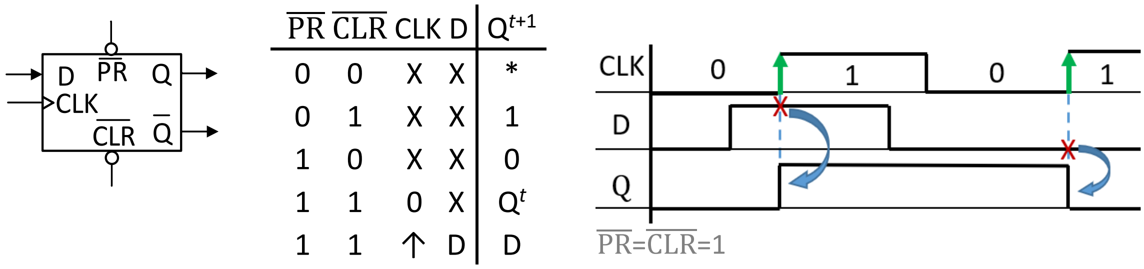


Figure 8. 4 The functionality of the D gated-latch triggered on the positive level

Note: There are D lathes triggered on the *negative level*, meaning input D is effective when CLK=0.

The TTL 7474 implements the D bistable with two additional asynchronous inputs for set ($\overline{\mathrm{PR}}$) and reset ($\overline{\mathrm{CLR}}$). The $\overline{\mathrm{PR}}$ (preset) and $\overline{\mathrm{CLR}}$ (clear) are active low. Their effect is immediate, and they have priority over the input D. When $\overline{\mathrm{PR}} = \overline{\mathrm{CLR}} = 1$ (disabled) , D is triggered on the rising edge of the clock. The rising edge takes place when CLK shifts from 0 to 1. Bistables with edge-based trigger are also called *flip-flops*. Level-based trigger bistables are called *latches*. The behavior of the flip-flop is presented in the timing diagram from Figure 8. 5. You can notice the effect of input D, only on the rising edge. The bistable reacts immediately after the edge.


D
$\overline{\mathrm{PR}}$
Q
CLK
$\overline{\mathrm{CLR}}$
$\overline{\mathrm{Q}}$
CLK
0
1
0
1
D
Q
$\overline{\mathrm{PR}}$=$\overline{\mathrm{CLR}}$=1

$\overline{\mathrm{PR}}$	$\overline{\mathrm{CLR}}$	CLK	D	Q^{t+1}
0	0	X	X	*
0	1	X	X	1
1	0	X	X	0
1	1	0	X	Q^t
1	1	↑	D	D

Figure 8. 5 The 7474 D flip-flop (left), the function table (middle) and its rising edge-based behavior (right)

Note: In Project Navigator, all TTL-based sequential logic circuits in the *ttl_env* project have an additional input, called CKF (ClocK-Fpga), which must be connected to the clock signal of the board, with a similar name. The clock signal of the board must be declared in the .ucf file, in the clock section, as follows:

```
## Clock signal
NET "CKF" LOC = "E3" | IOSTANDARD = "LVCMOS33";
NET "CKF" TNM_NET = sys_clk_pin;
TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 100 MHz HIGH 50%;
```

Important: The user clock on the test board will be generated using any of the 5 buttons. For each push, a button may generate several clock pulses due to mechanical deterioration (noise). The DECOUNCER unit (Figure 8. 6) available in the library of the *ttl_env* project, can be used to filter out false pulses. Its BT_IN input will be connected to the button terminal. The filtered signal will be available on BT_OUT. The CKF input will be connected to the clock signal of the board. The test circuit for TTL 7474 is highlighted next:

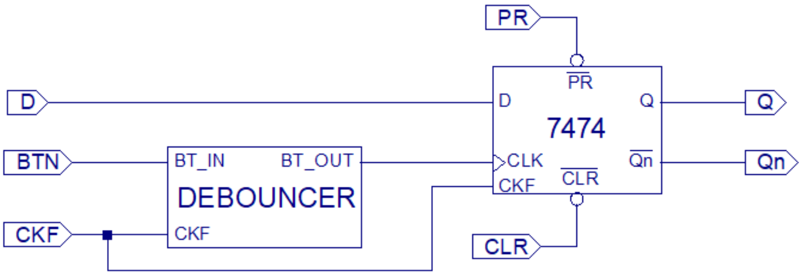


Figure 8. 6 The circuit for testing 7474 D flip-flop in Project Navigator

Note: The .ucf file contains a special section for declaring the buttons used:

```
## Buttons
NET "BTN" LOC=N17 | IOSTANDARD=LVCMOS33; # center
```

In Logisim, the signal CKF is missing, because it is specific only to FPGA boards. The DEBOUNCER unit is unnecessary because the simulator is noise-free.

Note: In practice, there are D also flip-flops triggered on the *falling edge* (on the clock shift from 1 to 0). Several solutions exist for implementing edge-based triggered bistables from level-based triggered bistables. One solution is to connect two SR latches in a *master-slave* configuration. The D input is connected to the master, and the Q output is generated by the slave. The implementation with NAND gates is highlighted in Figure 8. 7. The master has direct connection to the clock signal, while it is inverted for the slave, leading to an effect of alternate triggering for both latches.

Functionality: A command on input D affects the master while CLK=1, but its output cannot propagate to the slave, until CLK=0. The values for the signals are highlighted in Figure 8. 8. It can be noticed how blocking intervals alternate based on the value of the CLK, leading to the edge-based effect: only the most recent command before the falling edge will trigger the output, with results visible immediately after the edge.

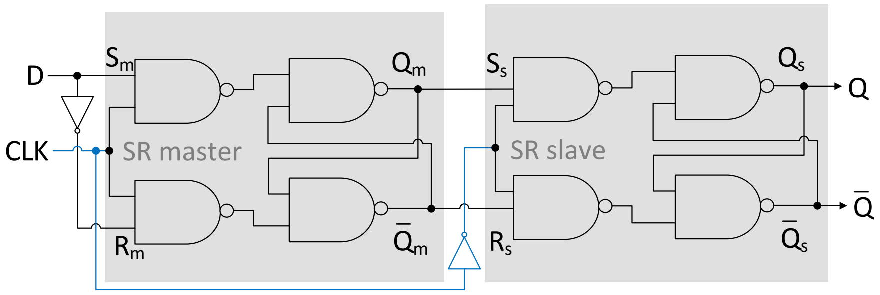


Figure 8. 7 Implementation of the master-slave D flip-flop using NAND gates

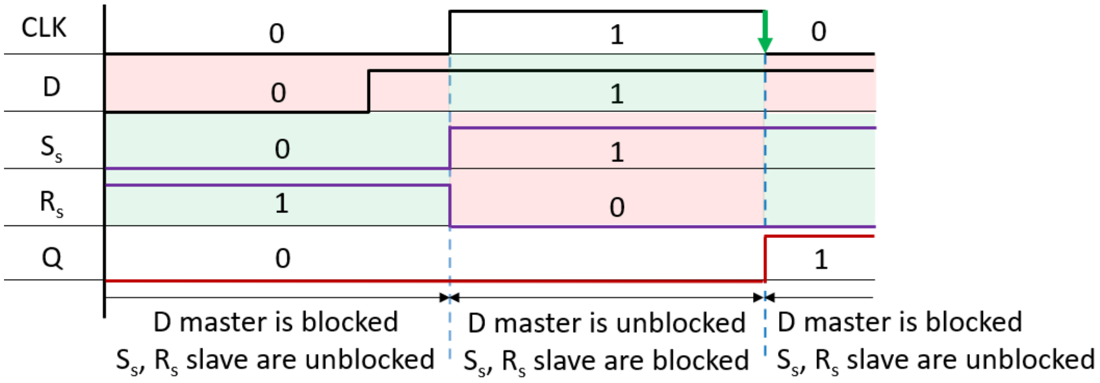


Figure 8. 8 The behavior of the signals from a master-slave D flip-flop

8.2.4 The JK gated-latch and JK flip-flop

The JK gated-latch has two synchronous inputs J and K, replacing the S and R inputs in a SR gated-latch. The functionality is almost like an SR circuit, except when J=K=1, the bistable state inverts itself. The function and the excitation tables are presented next:

Table 8. 3 The function table (left) and the excitation table (right) of the JK gated-latch

CLK	J	K	Q^{t+1}
0	X	X	Q^t
1	0	0	Q^t
1	0	1	0
1	1	0	1
1	1	1	$\overline{Q^t}$

Q^t	Q^{t+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

The NAND-based implementation extends the structure of an SR gated-latch with the following expressions for the inputs: $S = \overline{Q^t} \cdot J$, $R = Q^t \cdot K$ (Figure 8. 9 – left). Merging AND gates with NAND gates generates the 2-level logic circuit in Figure 8. 9 – right.

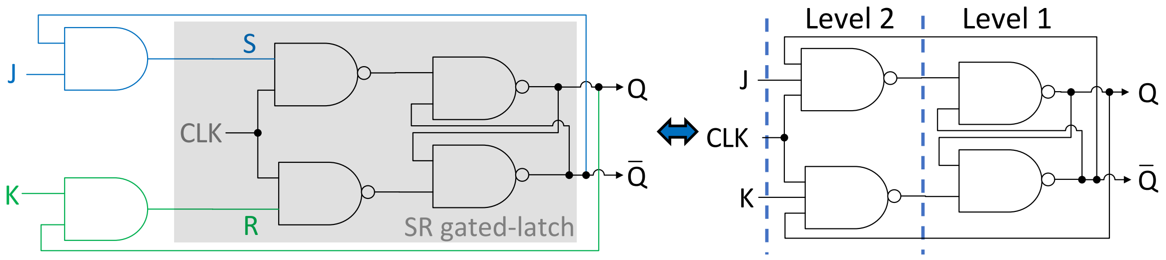


Figure 8. 9 The JK gated-latch implemented using basic logic gates

The master-slave version below of the JK flip-flop is triggered on the falling edge. The reactions are connected from the outputs of the slave to the inputs of the master:

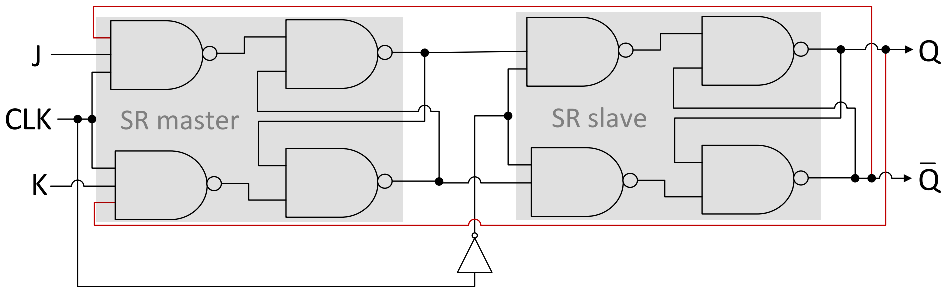


Figure 8. 10 The master-slave JK flip-flop implemented using NAND gates

The edge-based triggering advantage over level-based triggering is outlined when J=K=1, because the bistable state suffers only one change during a clock cycle (Figure 8. 11). For a level-based triggered JK latch, multiple changes are possible, depending on the duration of the clock period, which can lead to uncertain behavior.

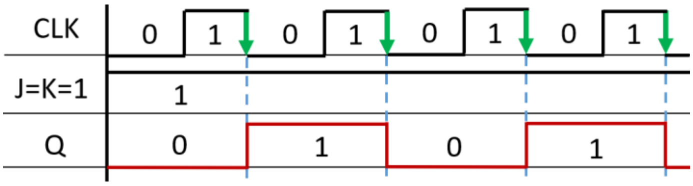


Figure 8. 11 The master-slave JK flip-flop state changes once per clock cycle

Circuits TTL 7473 and TTL 7476 implement master-slave JK flip-flops triggered on the falling edge. The 7473 has an asynchronous input $\overline{\mathrm{CLR}}$ (clear) for reset. The 7476 has both an asynchronous $\overline{\mathrm{CLR}}$ and an additional asynchronous set, called $\overline{\mathrm{PR}}$ (preset). The asynchronous inputs are active low and have priority over the synchronous inputs. The circuits are highlighted in Figure 8. 12.

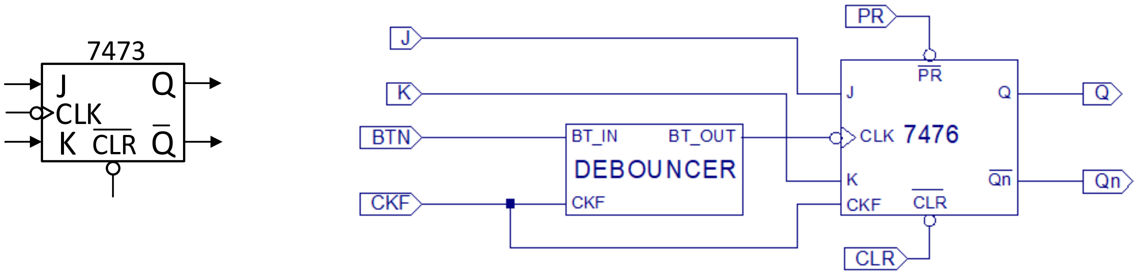
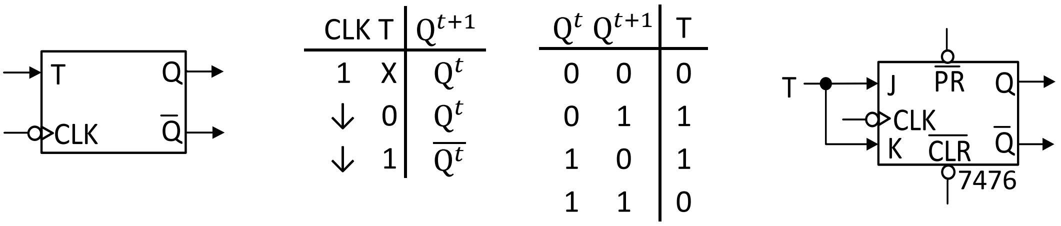


Figure 8. 12 The 7473 JK flip-flop (left) and the testing circuit for 7476 JK flip-flop (right)

8.2.5 The T gated-latch and T flip-flop (T ≈ toggle)

The T gated-latch has one synchronous input, called T. When T=0, the state is unchanged. The state toggles when T=1. There are also flip-flop variants of the T bistable. Its falling-edge triggered variant is presented in Figure 8. 13. Considering the function table, the *characteristic equation* is: $\mathrm{Q}^{t+1} = \mathrm{T} \oplus \mathrm{Q}^{t}$. **Note**: The behavior is identical to a JK bistable having J and K connected to T. The right side of Figure 8. 13 highlights the implementation of a T flip-flop using the 7476 JK flip-flop.



CLK	T	Q^{t+1}
1	X	Q^t
↓	0	Q^t
↓	1	$\overline{Q^t}$

Q^t	Q^{t+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Figure 8. 13 The T flip-flop triggered on the falling-edge (left), the function and excitation tables (middle) and its implementation using the 7476 JK flip-flop (right)

8.2.6 Implementing a bistable with other type of bistables

Implementing a bistable of type A with a bistable of type B is based on both excitation tables, as follows:

1. The pairs in both tables are associated based on similar values for the current and next states (Q^t, Q^{t+1}). A new truth table is generated, which places the current state Q^t and the type A bistable inputs on the left side, while having the type B bistable inputs on the right side.
2. The expressions of the type B bistable inputs are extracted using the values from the truth table.

Several examples of bistable “conversion” are presented next:

a) Implementing D using JK

When implementing a D using a JK we use their excitation tables:

Q^t	Q^{t+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Q^t	Q^{t+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Pairing common values for (Q^t, Q^{t+1}) in both tables we obtain the truth table below, which expresses the J, K inputs as functions of current state Q^t and input D. The expressions for J and K can be extracted using the Minimal Disjunctive Normal Form from the Karnaugh maps: $J = D$ and $K = \overline{D}$. We obtain the following implementation:

Q^t	D	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

$J = D$
$K = \overline{D}$

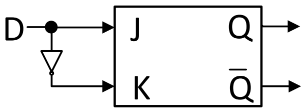
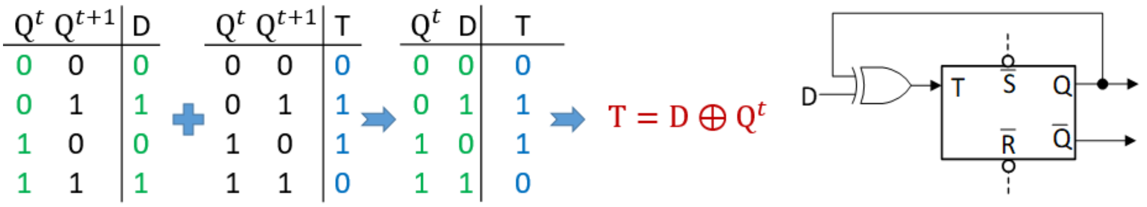


Figure 8. 14 Truth table for inputs J, K and the implementation using a bistable of type D

b) Implementing D using T

We use the excitation tables for D and T, and generate the truth table expressing T. Extracting the expression from the table we end up with the following solution:



Q^t	Q^{t+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Q^t	Q^{t+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

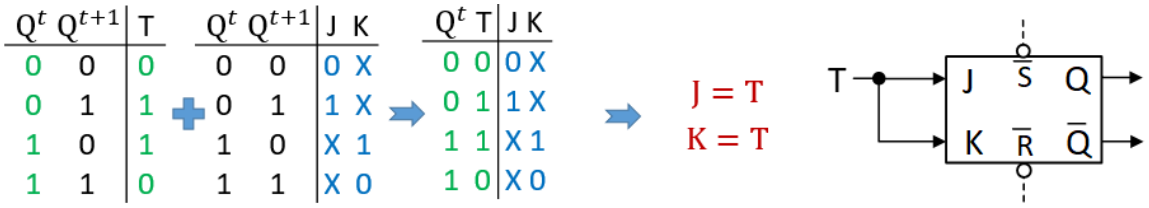
Q^t	D	T
0	0	0
0	1	1
1	0	1
1	1	0

$$T = D \oplus Q^t$$

Figure 8. 15 The steps for implementing a D bistable using a T bistable

c) Implementing T using JK

Using the excitation tables for T and JK we generate the truth table expressing the inputs J and K. Extracting the expressions from the table we obtain the following circuit:



Q^t	Q^{t+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Q^t	Q^{t+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Q^t	T	J	K
0	0	0	X
0	1	1	X
1	1	X	1
1	0	X	0

$$J = T$$
$$K = T$$

Figure 8. 16 The steps for implementing a T bistable using a JK bistable

8.3 Assignments

1. Implement using NAND gates and test on the board the asynchronous SR latch.
2. Implement on the board the 7474 D flip-flop. Test the asynchronous and synchronous commands.
3. Implement on the board the 7476 JK flip-flop. Test the asynchronous and synchronous commands.
4. Implement and test in Logisim the synchronous SR gated-latch using NAND gates.
5. Implement in Logisim a T flip-flop using the 7476 JK flip-flop. Test the asynchronous and synchronous commands.
6. Implement in Logisim the 7473 JK flip-flop. Test the asynchronous and synchronous commands.
7. Implement in Logisim a JK flip-flop using the 7474 D flip-flop. Disable the asynchronous commands, by connecting to VCC, and test the synchronous commands.