Module 2.2 - The SR Bi-stable

Aims of this module:

On successful completion of this module the learner can:

Describe Bi-stables (Flip-Flops) made from logic integrated circuits and can:
   a) Describe typical applications for SR Bi-stables
   b) Demonstrate an understanding of a) by compiling state tables for SR Bi-stables
   c) Recognize commercially available forms of integrated circuit SR Bi-stables in single and multiple forms.
   d) Recognize alternative forms of the SR Bi-stable
      i) Clocked SR Bi-stables
      ii) High Activated SR Bi-stables

Practical Activities connected with this Module

1. Using circuit simulation software, construct the circuits described.
2. For each simulation, derive a state table illustrating the circuit’s actions.
3. Recognize given standard circuit symbols for SR Bi-stables and from an understanding of their operation, complete given timing diagrams.

How you can assess your learning:

   By taking a multi-choice test.
   By completing the practical exercises included in this module.
Bi-stables

*Learning object a*) **Typical applications for SR Bi-stables**

The basic building block that makes computer memories possible, and is used in many sequential logic circuits is the Bi-stable or flip-flop circuit. Just two inter-connected logic gates make up the basic form of this circuit whose output has TWO STABLE OUTPUT STATES. The circuit can be triggered into either one of these states by a suitable input pulse and will “remember” that state until it is changed by another input pulse, or until power is removed. For this reason the circuit may also be called a Bi-stable Latch.

The SR Bi-stable can be considered as a 1-bit memory, since it "stores" the input pulse even after it has passed. Bi-stables of different types can be made from logic gates and as with other combinations of logic gates, the NAND and NOR gates are the most versatile. The NAND being most widely used because as well as being universal (it can be made to mimic any of the other standard logic functions) it is cheaper to construct. Other, more widely used types of Bi-stable are the JK, the D type, which are developments of the SR Bi-stable and will be studied in another part of this course.

**The SR Bi-stable.**

The SR (Set – Reset) Bi-stable is the simplest of all sequential circuits and is the basic building block from which all memory circuits, registers, digital counters etc. are constructed. It consists of two gates connected as shown in fig 1. Notice that the output of each gate is also connected one of the inputs of the other, giving a form of feedback or "cross coupling". The circuit has two inputs marked "not S" and "not R" ("NOT" being indicated by the bar above the letter) as well as two outputs, Q and "not Q".

![SR Bi-stable](image-url)
Bi-stables

*Learning object b*)

**State table for a SR Bi-stable**

Note: In combinational logic the table describing the true/false operation of a gate or circuit is called a truth table, but in sequential logic a similar table describes the state of the outputs after a particular input event or at a particular time and is called a state table.

The state table in fig 2 shows what happens to the Q and not Q outputs after a logic 0 is applied to either the not S or not R inputs.

A logic 0 at the not S input sets the Q output to logic 1 and a logic 0 at the not R input resets Q to 0. Not Q is, for these conditions, the inverse of Q.

The SR Bi-stable is a simple 1bit memory. If the set input is taken to logic 0 then back to logic 1, any further logic 0 pulses at this input will have no effect on the output. Thus a 0 pulse (high-low-high) at either input is "remembered" by the Q output being set or reset as appropriate.

You can check this and trace through the operation of the circuit by using the truth table for a single NAND gate to check the possible logic outputs. (Remembering that a change at either of the Q outputs changes the appropriate NAND gate input.)

Because of the feedback used in this circuit, tracing the action in this way is not quite straightforward. It is more useful to simply learn and remember the actions described in Fig 2.

**Fig. 2**

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Q is "Set" to 1 by logic 0 applied to S

No change - Circuit remembers 0 on S
<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Q is "Reset" to 0 by logic 0 applied to R
<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

No change – Circuit remembers 0 on R
<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

If both inputs are at logic 0 – both outputs will be logic 1 (not allowed!)
<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

If both inputs change from 0 to 1 at the same time, the outputs will be INDETERMINATE

A logic 0 at the not S input sets the Q output to logic 1 and a logic 0 at the not R input resets Q to 0. Not Q is, for these conditions, the inverse of Q.

The SR Bi-stable is a simple 1bit memory. If the set input is taken to logic 0 then back to logic 1, any further logic 0 pulses at this input will have no effect on the output. Thus a 0 pulse (high-low-high) at either input is "remembered" by the Q output being set or reset as appropriate.

You can check this and trace through the operation of the circuit by using the truth table for a single NAND gate to check the possible logic outputs. (Remembering that a change at either of the Q outputs changes the appropriate NAND gate input.)

Because of the feedback used in this circuit, tracing the action in this way is not quite straightforward. It is more useful to simply learn and remember the actions described in Fig 2.

**Fig. 3 The NAND gate and its truth table.**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Bi-stables

Learning object c) Multiple SR Bi-stables

Although these are useful circuits found in many digital systems, there are few commercially available packages containing multiple SR Bi-stables. The reason for this is firstly that some of the more complex and versatile Bi-stables can duplicate the action of and SR Bi-stable, so there is not a great need for both. Secondly the circuit is very simple to make from two NAND gates.

However Texas instruments list data for one (now obsolete) multiple package containing four SR Bi-stables, the SN74279 at [http://focus.ti.com/lit/ds/symlink/sn74279.pdf](http://focus.ti.com/lit/ds/symlink/sn74279.pdf)
Bi-stables

Learning object d)

i) High activated SR Bi-stables

Sometimes it is desirable to have a SR Bi-stable with high-activated inputs. This can be achieved as shown in fig 4a by using two additional NAND gates connected as inverters, to invert the inputs.

An alternative method is to use NOR gates instead of NAND. This gives a simpler circuit, but notice that when using NOR gates to construct a high-activated bistable, the Q and not Q outputs are switched around; not Q is now the output from gate X.

ii) The Clocked SR Bi-stable

Fig 5 shows a further development of the SR Bi-stable. Here the two inverters at the inputs are wired so that the input states at the S or R inputs will only appear (inverted) at the not S or not R Bi-stable inputs when a logic 1 is present at the CK (clock) input. Thus we can arrange that the Q output will only be set when both the S input and the CK input are high, otherwise the S input is ignored. Likewise the R input is only active when the CK input is high. This arrangement would make a better 1 bit memory device by using the S or R inputs for data while using the CK input to ensure that one bit of data was stored only in this particular Bi-stable out of many similar Bi-stables by making only this Bi-stable’s CK input high at the time the data is to be stored.

Fig 6 Circuit Symbols for SR Bi-stables
Example Practical Exercise for module 2.2

The two circuits shown below were constructed in “Circuit Maker” (Student Edition), which is available as a free download [here](#). This software is typical of a number of free design/simulation programs available as Internet downloads allowing the design simple analogue or digital circuits on screen. Simulations can then be run to test their operation.

1. Construct each of these circuits using Circuit Maker and use them to compile a state table showing their operation.

![Circuit Simulations constructed in “Circuit Maker”](#)

**Explanation**

The circuit on the left operates correctly; the one on the right is designed to show the non-allowed and indeterminate states when the inputs change from 00 to 11 at exactly the same time.

When you run the simulation with the right hand circuit, the indeterminate state is shown by rapidly changing outputs. In a real circuit the outputs would not continually change, as in the simulation; they would be different from each other, but we would not be able to reliably forecast which of the outputs would go to logic 1 and which to logic 0.

The SR Bi-stable has some major drawbacks however. Look at the state table in fig 2 again and notice that if both inputs are at logic 0 together, then both Q and not Q are logic 1. So in this state not Q is not the inverse of Q, it is identical! Clearly an undesirable state of affairs, but even worse is to come.....

Look at what happens if we allow both inputs to change from 00 to 11 together. The states of the two outputs cannot be guaranteed. They are said to be "indeterminate". In fact, which output goes to logic 1 will depend on the gains of the transistors within the individual gates. One gate will usually turn on before the other forcing a particular output, but which gate "wins the race" to logic 1 will be a matter of chance. Both of these undesirable states just described must be overcome. In the simple SR bistable the way this is usually done is simply to avoid the 00 input state. More complex bistables have better ways of avoiding these undesirable states.

2. Construct and test circuits for:
   a) High activated SR Bi-stables from
      i) NAND gates
      ii) NOR gates.
   b) A Clocked SR Bi-stable from NAND gates

3. For each circuit, draw a state table showing the operation of the circuit.
Example Problem for Module 2.2.

Describe the action of a Bi-stable using a timing diagram.

A timing diagram is an alternative way of illustrating the operation of a digital circuit. It shows how the logic levels at inputs vary with time, and how the corresponding outputs change in response.

**Problem:** For the device shown; complete the timing diagram for outputs $Q$ and $Q'$.
Model answer for Problem 2.2 Describe the action of a Bi-stable

For the device shown; complete the timing diagram for outputs Q and Q

Timing Diagram

(Not part of Student materials)
Sample multi-choice questions relating to module 2.2

1. A sequential logic circuit consisting of a combination of logic gates that produces only TWO stable output conditions is called;
   a. A monostable.
   b. An astable.
   c. A bistable.
   d. A multivibrator.

2. Refer to fig 204. The circuit shown is;
   a. A register.
   b. A bistable latch.
   c. A clock oscillator.
   d. A pulse generator.

3. In a simple low activated SR flip flop, the outputs Q and not Q will be INDETERMINATE (unknown) if;
   a. Both not S and not R inputs are at logic 0 together.
   b. Both not S and not R inputs go from logic 0 to logic 1 together.
   c. Both not S and not R inputs are at logic 1 together.
   d. The reset input is 0.

4. Refer to fig 205. If the circuit shown, while in the RESET condition, has the indicated inputs applied, the outputs will be;
   a. Q = 1 not Q = 0
   b. Q = 0 not Q = 0
   c. Q = 0 not Q = 1
   d. Q and not Q are indeterminate.