Chapter 6

Ring oscillators and multi-stable circuits

6.1 Ring oscillators

Suppose we take five inverters and connect them end to end as shown in Figure 6.1.

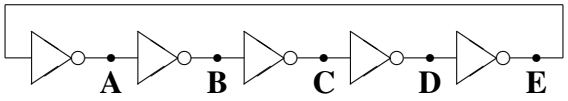


Figure 6.1: Five stage ring oscillator.

Let us assume that node **A** in the circuit is 0. The second inverter will drive node **B** to 1. Then the third inverter will drive node **C** to 0, the fourth inverter will drive node **D** to 1 and the fifth inverter will drive node **E** to 0. This in turn will make the first inverter drive node **A** to 1, which is contrary to the original assumption. If we had assumed instead that node **A** was 1, we would have found that the first inverter drives it to 0.

The key is that when a signal passes through an inverter (or any logic gate for that matter), it not only is inverted, it is also delayed slightly. Hence, if node **A** starts out being 0, after five inverter gate delays, it will be driven to 1, and then after another five inverter gate delays, it will be driven to 0.

What we have constructed is an oscillator. The oscillation period is the time it takes for a signal to go around the loop twice (once while the output is 0 and once while it is 1). Hence, for an n stage ring oscillator, the oscillation frequency is given by:

$$F_{\rm osc} = \frac{1}{2nt_{\rm pd}} \quad \text{for } n \text{ odd}$$

where $t_{\rm pd}$ is the inverter delay time or *propagation delay*.

Ring oscillators are often used to characterize the speed of an integrated circuit manufacturing process. For this application, long rings of typically 31 stages are used. This results in oscillation frequencies of around 100 MHz or higher.

If the number of stages is reduced to three, the circuit still oscillates in theory. However, in practice, ring oscillators with small numbers of stages (3, 5 and 7) may not oscillate at all or the oscillations may be too small in amplitude to be useful.

If the number of stages is reduced to one, the single inverter will not oscillate, and will partially short circuit the power supply. That is, a large amount of current will flow from Vdd to Gnd, but not quite as large as in the case Figure 2.6(e), and the output of the inverter will be approximately halfway between Vdd and Gnd.

6.2 Bistable circuits

If the number of stages in the ring is even, we find that we can start by assuming a particular node is either 0 or 1 without ending with a contradiction. This is because the resulting circuit is bistable, and does not oscillate.

The simplest instance of such a circuit is the two stage ring shown in Figure 6.2. This circuit can exist in either of two states. Either node **A** is 0 and node **B** is 1 or node **A** is 1 and node **B** is 0. Theoretically, the state that the circuit is in will be randomly selected when the circuit is first powered up, but in practice, the design is unlikely to be perfectly symmetric, so that the circuit will tend to prefer one state over the other.

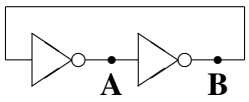
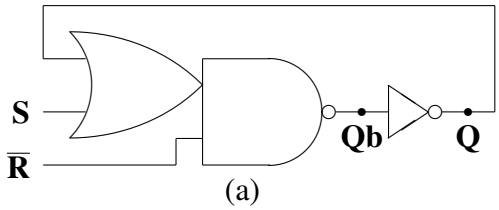


Figure 6.2: Two stage bistable ring.

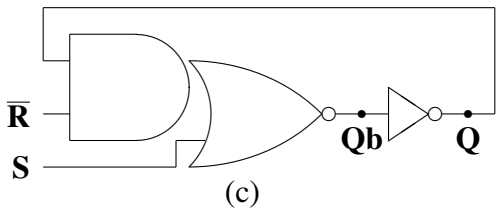
The usefulness of this circuit is questionable. However, we can modify it to allow for setting the state. By adding a set and a reset function to the first inverter as described in Figures 5.7(b) and (c), the inverter becomes an OR-NAND gate or an AND-NOR gate. This is shown in Figures 6.3(a) and (b).



(a)

S	$\overline{R}$	Q	Qb
0	0	0	1
0	1	**Q**	**Qb**
1	0	0	1
1	1	1	0

(b)



(c)

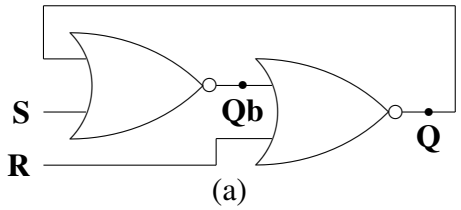
S	$\overline{R}$	Q	Qb
0	0	0	1
0	1	**Q**	**Qb**
1	0	1	0
1	1	1	0

(d)

Figure 6.3: Set-reset flip-flops. (a) OR-NAND implementation. (b) Truth table for OR-NAND implementation. (c) AND-NOR implementation. (d) Truth table for AND-NOR implementation.

Alternatively, we can add a set or a reset function to both inverters, converting them both to either a NAND gate or a NOR gate, as shown in Figures 6.4(a) and (b). The disadvantage of this

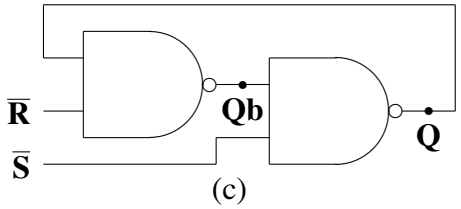
method is that neither the set nor the reset function has a clear priority. Although no short circuit occurs when both functions are active, both gate outputs go to 0 (for the NOR gates) or 1 (for the NAND gates).



(a)

S	R	Q	Qb
0	0	**Q**	**Qb**
0	1	0	1
1	0	1	0
1	1	0	0

(b)



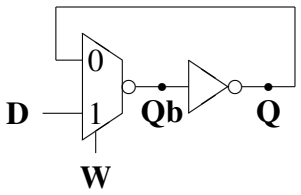
(c)

$\overline{S}$	$\overline{R}$	Q	Qb
0	0	1	1
0	1	1	0
1	0	0	1
1	1	**Q**	**Qb**

(d)

Figure 6.4: Symmetric set-reset flip-flops. (a) NOR implementation. (b) Truth table for NOR implementation. (c) NAND implementation. (b) Truth table for NAND implementation.

Finally, if we want to be able to set the state of the circuit to a value given by another signal, we can replace one of the inverters with the inverting multiplexor from Figure 3.5 as shown in Figure 6.5.



(a)

W	D	Q	Qb
0	0	**Q**	**Qb**
0	1	**Q**	**Qb**
1	0	0	1
1	1	1	0

(b)

Figure 6.5: D flip-flop. (a) Implementation. (b) Truth table.

6.3 Multi-stable circuits

In some instances, we would like to have a circuit that has more than two stable states. For instance, a circuit with three stable states can be constructed using three logic gates instead of two inverters. The two possible configurations are shown in Figure 6.6. When constructed with NAND gates, only one of the three outputs can be 0 at any given time, and when constructed with NOR gates, only one of the three outputs can be a 1 at any given time. These tri-stable circuits can be modified in the same way as the bistable circuit of Figure 6.2 to provide for set and reset functions.

This topology can be extended to more than three stable states, but the number of MOSFETs used increases quadratically with the number of states. If on the other hand we use multiple

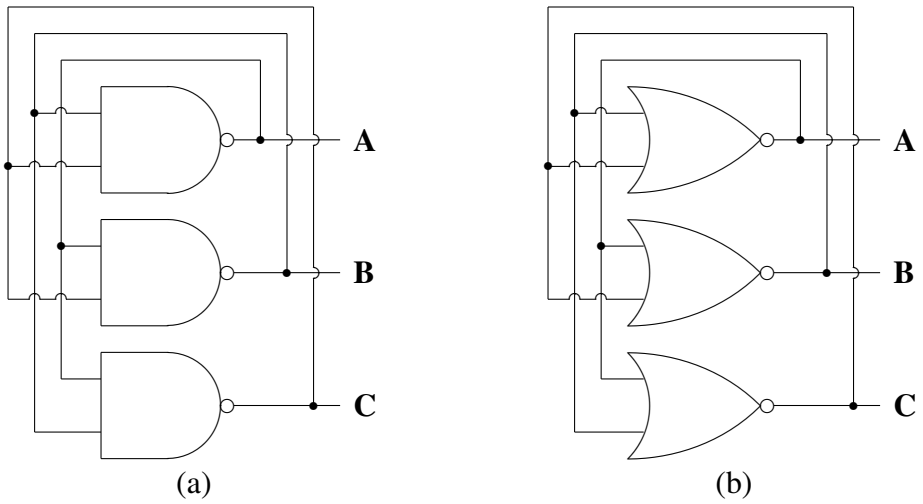


Figure 6.6: Tri-stable circuits.

independent bistable circuits, the number of MOSFETs used increases only logarithmically with the number of states.

However, in some critical cases, it is necessary to ensure that the circuit powers up in a valid state. For instance, if we need only three states, the circuit in Figure 6.6 can only power up in one of those states. If we use two bistable circuits, the circuit can power up in one of the three valid states or the fourth invalid one. If the fourth state has a risk of causing a short circuit (because it directly controls a logic gate like the one in Figure 5.7(a)), then the additional MOSFETs required to prevent this occurrence may exceed the number of MOSFETs used in Figure 6.6.

6.4 Clocked latches

Synchronous sequential circuits require the use of many clocked data latches to hold the result of the previous computation while the current computation is performed.

We cannot use the D flip-flop of Figure 6.5 alone because when we are writing to it ($\mathbf{W} = 1$), the circuit is transparent – that is, **Q** immediately reflects whatever changes occur to **D**. Hence, using it in a configuration with feedback would produce a ring which would most likely oscillate. The solution is to use a master-slave configuration with two of them in series. This is shown in Figure 6.7.

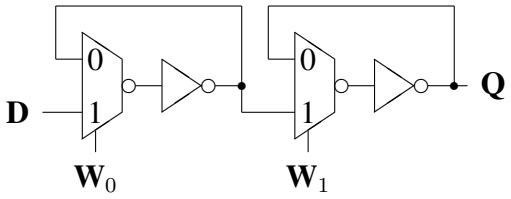


Figure 6.7: Master-slave data latch configuration.

In order to avoid simultaneous transparent conditions for the master-slave configuration, we need to ensure that $\mathbf{W}_0$ and $\mathbf{W}_1$ are not 1 at the same time. Inverting $\mathbf{W}_0$ to produce $\mathbf{W}_1$ is not

enough, because the propagation delay through an inverter is not zero. Hence, on the falling edge of $\mathbf{W}_0$, $\overline{\mathbf{W}_0}$ remains high for time $t_{\rm pd}$ as shown in Figure 6.8. Modifying the second D flip-flop so that the feedback to the multiplexor is on the 1 input is also not enough, because the multiplexor itself needs an inverted copy of its select signal anyway (see Figure 3.5).

The solution is to use a two-phase non-overlapping clock scheme. The timing for this type of clocking is shown in Figure 6.8 (with an exaggerated $t_{\rm pd}$ for generating the complementary signals). In this scheme, there is sufficient time between $\mathbf{W}_0$ pulsing high and $\mathbf{W}_1$ pulsing high to guarantee that even with the $t_{\rm pd}$ required to invert each of them, there is no overlap.

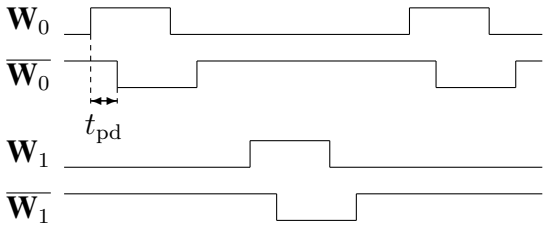


Figure 6.8: Timing of two-phase non-overlapping clock signals for master-slave data latch.

6.5 Generating two-phase non-overlapping clocks

Oscillators and other clock sources usually do not generate two-phase non-overlapping clocks directly. The conventional way to generate two-phase non-overlapping clocks from a single clock is to start with one of the symmetric set-reset flip-flops of Figure 6.4, and add an inverter between the set and reset inputs as shown in Figure 6.9(a). The behavior of this circuit is sketched in Figure 6.9(b).

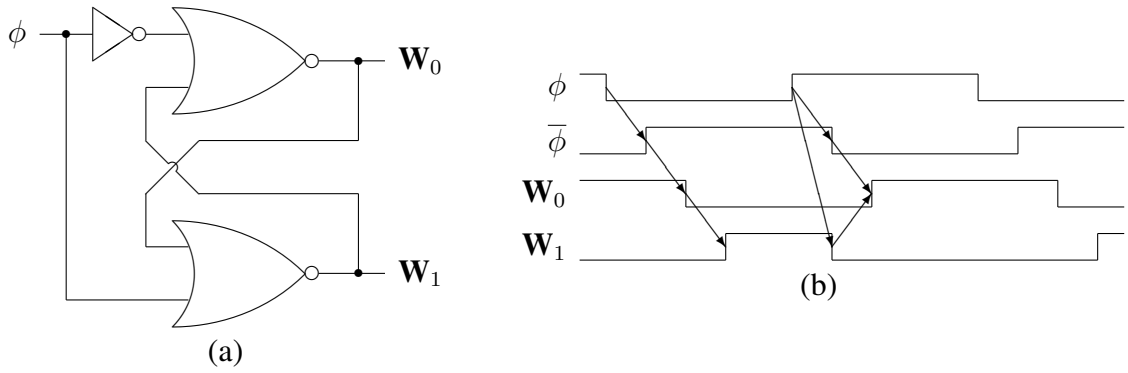


Figure 6.9: Two-phase non-overlapping clock generation.

The operation of the circuit is as follows. When the input clock ϕ goes low, $\overline{\phi}$ goes high after a delay due to the inverter's propagation delay. This in turn causes $\mathbf{W}_0$ to go low after a delay due to the upper NOR gate's propagation delay. Finally, this causes $\mathbf{W}_1$ to go high after a delay due to the lower NOR gate's propagation delay.

When the input clock ϕ goes high, this simultaneously causes $\overline{\phi}$ goes low after a delay due to the inverter's propagation delay and $\mathbf{W}_1$ to go low after a delay due to the lower NOR gate's

propagation delay. Once $\overline{\phi}$ and $\mathbf{W}_1$ are both low, $\mathbf{W}_0$ will go high after a delay due to the upper NOR gate's propagation delay.

Note that while $\mathbf{W}_0$ is high for as long as it is low (assuming a symmetric input clock ϕ), $\mathbf{W}_1$ is high for a shorter amount of time than it is low. Nevertheless, the operation of the set-reset flip-flop prevents $\mathbf{W}_0$ and $\mathbf{W}_1$ from being high simultaneously, which is what we need for our two-phase non-overlapping clock generation. If we use NAND gates instead of NOR gates, we produce similar two-phase non-overlapping clocks that are never simultaneously low.

The principal problem with this circuit is that the duration of the non-overlapping region is determined by the propagation delay of the NOR gates. Since $\mathbf{W}_0$ and $\mathbf{W}_1$ need to be inverted to drive the D flip-flops, this may be insufficient. We can increase the duration of the non-overlapping region by adding inverters as shown in Figure 6.10.

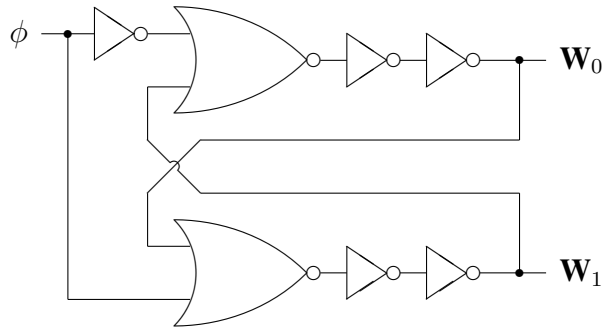


Figure 6.10: Two-phase non-overlapping clock generation with extended non-overlapping region.

However, suppose that after the integrated circuit is fabricated, we discover that the synchronous circuit driven by this two-phase non-overlapping clock generator is slower than expected and that the duration of the non-overlapping region is insufficient for proper operation. We can no longer change the number of extra inverters in the clock generator, and we cannot affect the duration of the non-overlapping region by changing the input clock frequency.

A better solution is to use a single clock toggle flip-flop and two extra NOR gates as shown in Figure 6.11. The single clock toggle flip-flop divides the input clock frequency by two, and the NOR gates produce the non-overlapping clocks on alternate periods of the input clock. The duration of the non-overlapping region is controlled by the duty cycle of the input clock, and the frequency of the generated clocks is one half the frequency of the input clock.

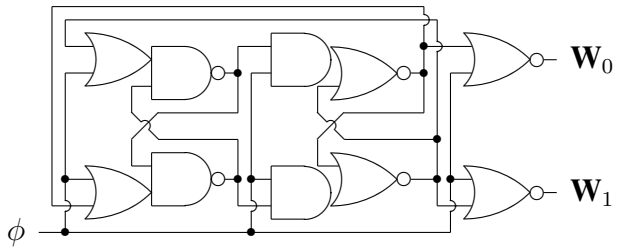


Figure 6.11: Two-phase non-overlapping clock generation using a single clock toggle flip-flop.