LATCHES, FLIP-FLOPS, AND REGISTERS

OBJECTIVE: DESIGN *4 word SRAM memory. EACH word is 16 BIT*
Tools: VHDL, **Model-SIM , Quartus, DE 2-70**
Due Date: March 14, 2015

\

Introduction

Latches are the basic circuits in constructing storage elements. In fact, it is *the* most basic storage element. A storage element can maintain a binary state until an input binary value signals it to change states. There are various types of latches, which in turn create various types of storage elements.

Another storage element this lab will concentrate on is the flip-flop. Flip-flops are simple circuits that can store a logic state until it needs to be used at a later time. There are many variations of flip-flops. This lab will concentrate on the D flip-flop.

Finally, this lab will give an example of a *register.* Registers are made up of several storage elements. Although registers can be made up of several latches, they are usually made up of several flip-flops. Each flip-flop stores 1 bit of information. Therefore, *n* flip-flops can store *n* bits of information. Further detail will be given in later sections.

SR-Latch

An SR-latch has 2 inputs, and 2 outputs. The inputs are called “Set” and “Reset”. The outputs are called “Q” and “notQ”. A logic diagram for this is shown below.

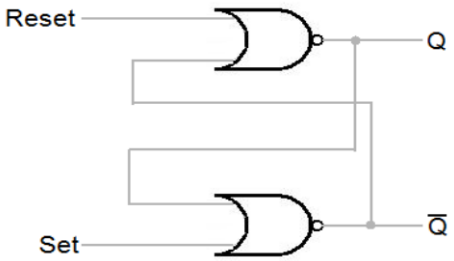

Figure 1: SR_Latch.

As figure 1 shows, only two NOR gates are needed to make a basic SR-latch. The feedback from the output of Q to the input of the bottom NOR gate, as well as the other feedback from notQ to the input of the top NOR gate, might be confusing. You do not need to worry about why this works right now. All you need to know is the logic behind it. A logic table is shown below to show you the logic.

S	R	Q	Not Q	State
1	0	1	0	Set State
0	0	1	0	Set State
0	1	0	1	Reset State
0	0	0	1	Reset State
1	1	0	0	Undefined

Table 1: Logic table for SR-latch.

The top set of values (S=1, R=0, Q=1, notQ=0) represent the “Set” state. This is because S=1 while R=0. However, in the next set of value, S goes to 0, and yet we are still in the Set state. This is because the states do not change unless both the set state goes to 0 *and* the Reset state goes to 1, as you can see in the third set set of value. The fourth set of values (S=0, R=0, Q=0, notQ=1) are still in the reset state. Again, although R went to 0, S did not go to 1, so that state remains “Reset”.

The final set of values can be confusing. In this design, when both S and R have a logic value of 1 at the same time, the value at that state becomes undefined. This also results in an unpredictable “next state”. This is because one of the requirements of this circuit is the outputs always be the compliment of each other.

SR-Latch With Control Input

The SR-latch with control input is very similar to the basic SR-latch, but with one extra input called “Control”. To make things a little simpler, assume we do not care about the value of notQ. This notQ output certainly exists in the design, but we will only care about the value of Q. A logic diagram is shown below in figure 2.

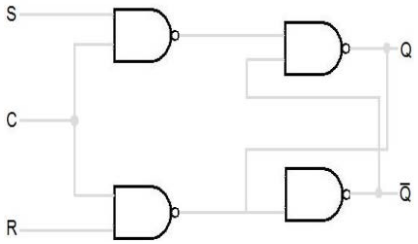

Figure 2: SR-latch with control input.

Figure 2 shows that this design uses 4 NAND gates. The extra control input determines if a change to the output Q will be made. This means that if C=0, the value of Q will not change regardless of the values of S and R. A logic table is shown below.

C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q=0; Reset state
1	1	0	Q=1; Set state
1	1	1	Undefined

Table 2: Logic table for SR-latch with control input.

You will need to use signals to make the design work properly. This is because an output cannot be used as an input.

D-Latch

The D-latch eliminates the possibility of having both S and R equal to 1 at the same time. In fact, there are no S and R inputs. The only two inputs in the D-latch are the control (or clock) input, and D. The "D" input will replace both S and R. The logic diagram below shows how this D input eliminates the undesirable S=1, R=1 scenario.

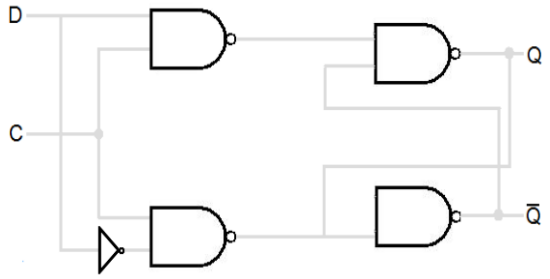

Figure 3: D-Latch.

As you can see, the D input is fed into the same places S and R would have been fed into. The NOT gate going into the bottom-left NAND gate prevents D from being a logic value of 1 in both the top-left and bottom-left NAND gates. A logic table for this design is shown below in table 3.

C	D	Next state of Q
0	X	No change
1	0	Q=0; Reset state
1	1	Q=1; Set state

Table 3: Logic table of D-latch.

Master-Slave D Flip-Flop

The master-slave flip-flop is made up of two latches. In the case of a master-slave D flip-flop, it will be made up of two D-latches. Depending on the value of the clock input, either the “master” or the “slave” latch will be activated (able to change value). A logic diagram of a master-slave D flip-flop is shown below.

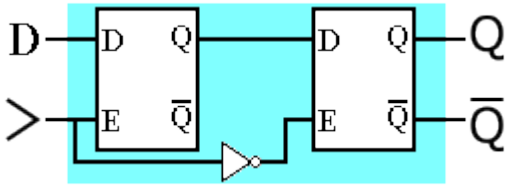

Figure 4: Master-slave D flip-flop.

It is important to note that this is a negative *edge-triggered* circuit, as opposed to the *pulse-triggered* design of the latches. This means that the output Q will equal the input D only on a clock edge (in this case, a rising edge). Therefore, Q can only change at the moment the clock goes from 1 to 0.

T Flip-Flop With Enable

Here's a T flip-flop created from a positive-edge triggered D Flip-flop.

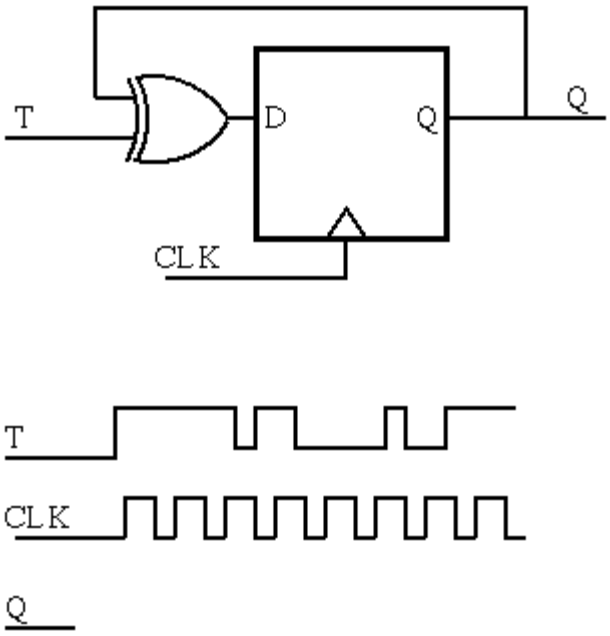

Figure 5: T flip-flop

Here's a JK Flip flop.

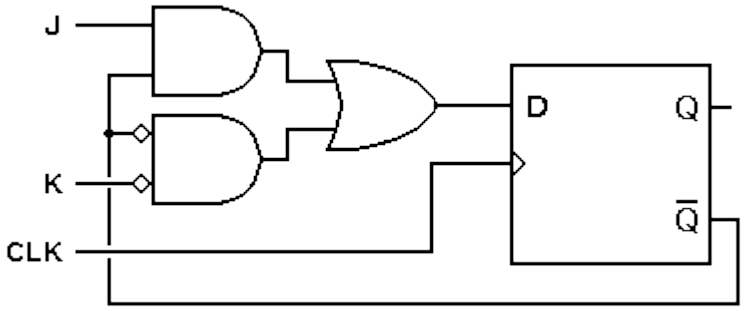

Figure 6: JK Flip-Flop

SRAM - (Registers)

Extend this section Now that you have created flip-flops, you can finally create a register. In this example, a 4-bit register with *parallel load* is to be created.

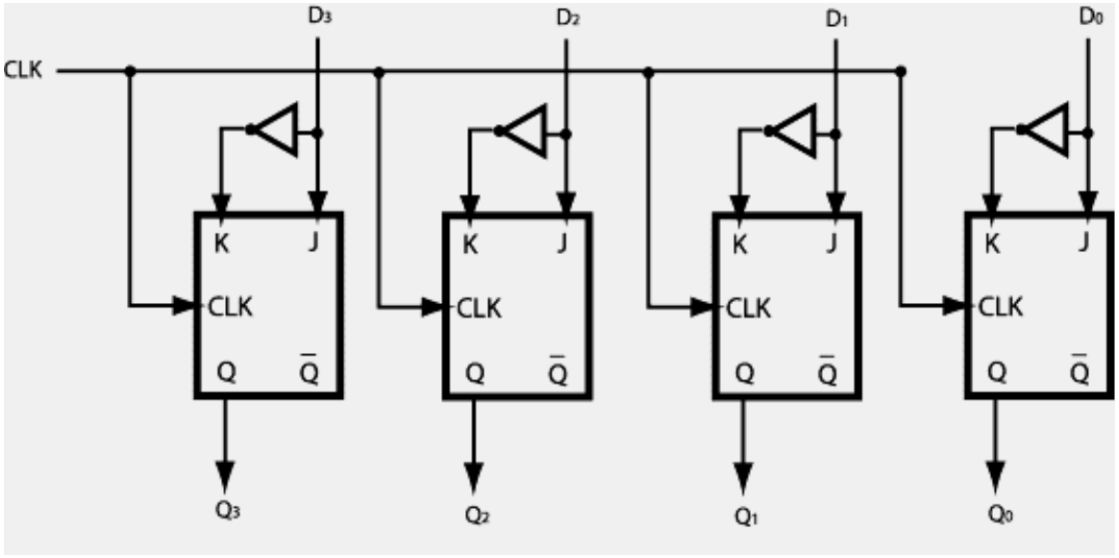

Figure 7: 4-bit Register.