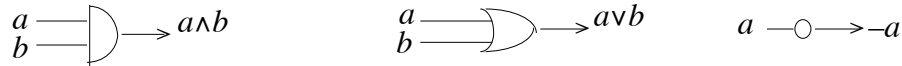


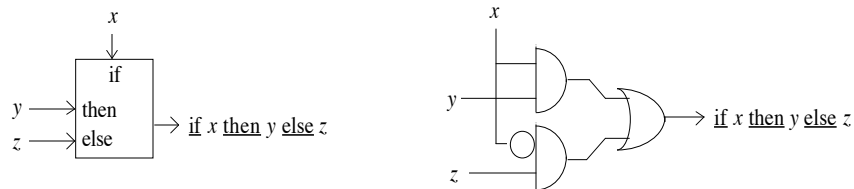
CSC258 Lecture Notes Circuit Design

Diagrams

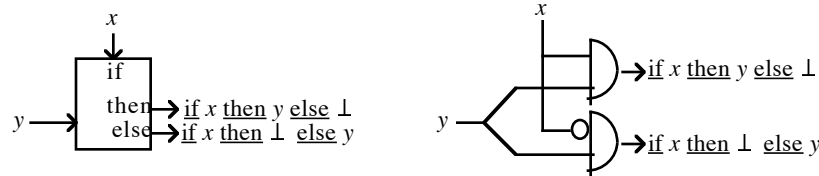
Circuits are often expressed as diagrams. The standard diagrams for \wedge , \vee , and \neg are:



(Textbooks use a triangle plus a circle for negation. We will use a triangle for something else.) We can give a diagram to any other operator by the expedient method of placing its symbol in a box. Here is the **if then else** (textbook term: multiplexer). On the right we show an implementation using \wedge , \vee , and \neg .

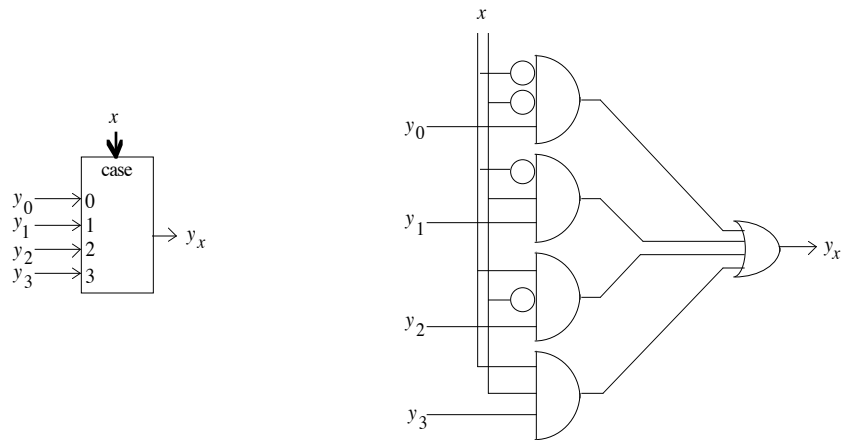


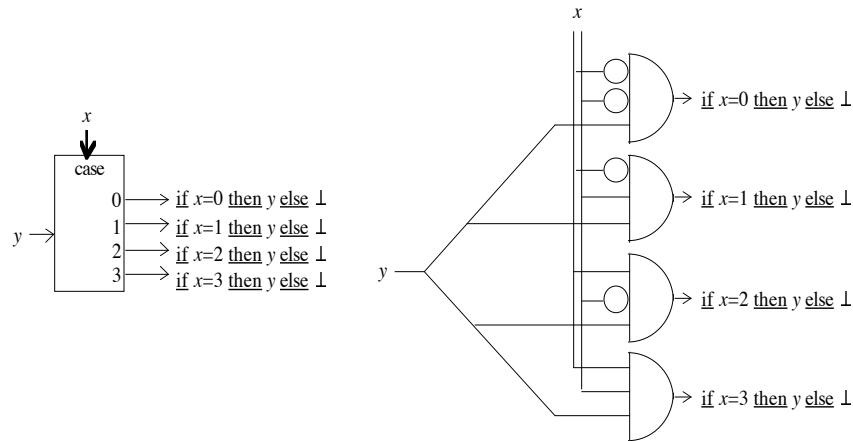
It is also convenient to define the switch (textbook term: demultiplexer), and to give it a diagram.



The symbol \perp is for “low voltage” or “ground” or “false”. We use \top for “high voltage” or “power” or “true”.

The **if then else** and switch also come in a more general form in which the selection is made by a natural rather than a binary. For a 2-bit natural selection the diagrams are



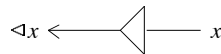


A thin line indicates a single wire, and a thick line indicates any number of wires. Crossing wires are not connected.

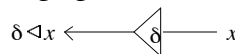
Time

Ideally, we might suppose that circuit components act instantly, with no gate delays, and are represented accurately by timeless binary expressions. Realistically, there are gate delays, and sometimes there are transient signals (glitches) while a circuit settles into a stable state. We must introduce a timing discipline to ensure that we do not require, and are not affected by, a result before it is ready. We can consider time to be continuous or discrete; nothing in these lecture notes will depend on that choice.

To talk about time, we find it convenient to introduce the operator \triangleleft , pronounced “delay” or “previous”. It gives the value that its operand had previously, a short time ago. Its diagram looks like this:



Whenever we need to say formally what constraints a delay time must satisfy, we write it to the left of the delay operator, and inside its circuit graphic:



Delay time is dependent on context and technology, it is usually determined by experiment, and can be known only approximately, say with an upper and lower bound. Sometimes we want the delay to be as short as possible; when that is the case, signal propagation time through the wire and surrounding gates is sufficient, and no extra circuitry is required. When more delay is needed, it can be implemented as an even number of negations, or by a suitable choice of layout; these implementations are not subject to glitches, and so do not raise again the problem they are solving. In addition to its logical use, the delay sometimes has the electrical job of reshaping a pulse, both height and width, to compensate for degradation. But that is a level of detail below our concern in these lecture notes.

As a formal requirement, for proof of correctness, we need to define the output of a delay to be initially \perp for the delay time, and thereafter it is the same as the input but delayed. This initial \perp is the only initialization in our circuits. (We don't consider initialization circuitry or proof of correctness in these lecture notes.)

Flip-flops

A flip-flop has inputs C (clock) and D (data), and output Q (the letter Q is like the letter O for output, but distinguishable from the digit 0). A flip-flop behaves as follows:

If the clock is T , then the output is the data input, otherwise the output remains as it was.

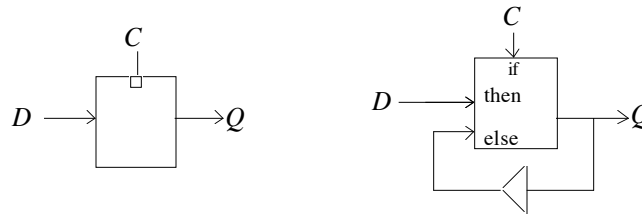
This behavior can be formalized directly as follows.

$$\text{if } C=T \text{ then } Q=D \text{ else } Q=\triangleleft Q$$

We can simplify this binary expression in two ways. Equating to T is always superfluous, just as adding zero or multiplying by one are superfluous. And we can factor out the “ $Q=$ ”, obtaining

$$Q = \text{if } C \text{ then } D \text{ else } \triangleleft Q$$

which we might well have written in the first place. We now have a definitional equation, which is the form suitable for automatic circuit fabrication. An equation is “definitional” if one side is the output. Here are the black box and implementation diagrams.



For the circuit to operate correctly, the delay must be small enough that no change to D occurs during the delay preceding the fall of the clock. For the circuit to be as fast as possible, the delay must be as small as possible. There will be a constraint on the minimum delay for electrical reasons, which we do not consider here. With no delay, this circuit is logically equivalent to a standard textbook flip-flop.

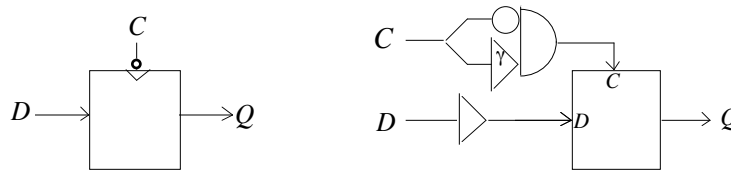
Edge-triggering

The flip-flops we have just described remain sensitive to the data input as long as the clock is \uparrow (textbook term: level-sensitive). Sometimes we want a flip-flop that is sensitive to its data input only at the rising or falling edge of the clock input. For example, a falling-edge-triggered flip-flop (textbook term: negative-edge-triggered) can be defined as

$$Q = \text{if } -C \wedge \gamma \triangleleft C \text{ then } \triangleleft D \text{ else } \triangleleft Q$$

$\gamma \geq (\text{edge time}) + (\text{negation delay})$

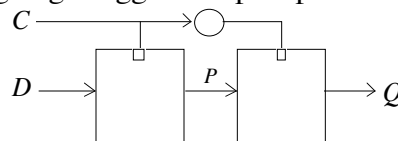
The expression $-C \wedge \triangleleft C$ says that the clock is down but was just previously up, so it is a falling edge. The C -delay γ should be just large enough to allow C to fall and to allow that falling edge to be negated. The D -delay determines what data is latched; for example, we might want the data from before the falling edge, or at its start, or at its end (this delay could be omitted). As always, the Q -delay should be as small as possible. In the box diagram (below left), the small triangle indicates an edge-triggered clock input, and the small circle indicates that it is falling-edge-triggered. For rising-edge-triggered, omit the small circle.



A logically equivalent, slightly larger, slightly faster circuit is obtained from

$$Q = (-C \wedge \triangleleft C \wedge \triangleleft D) \vee (-\triangleleft C \wedge \triangleleft Q) \vee (C \wedge \triangleleft Q)$$

Textbooks obtain an edge-triggered flip-flop with a master-slave pair of level-sensitive flip-flops. For example, they obtain a falling-edge-triggered flip-flop as follows:



Algebraically, this is

$$P = \text{if } C \text{ then } D \text{ else } \triangleleft P$$

$$Q = \text{if } -C \text{ then } P \text{ else } \triangleleft Q$$

We now prove that this master-slave pair is equivalent to our single flip-flop. Apply a delay to the P equation and to the Q equation to obtain

$$\triangleleft P = \text{if } \triangleleft C \text{ then } \triangleleft D \text{ else } \triangleleft \triangleleft P$$

$$\triangleleft Q = \text{if } -\triangleleft C \text{ then } \triangleleft P \text{ else } \triangleleft \triangleleft Q$$

From the P , Q , and $\triangleleft P$ equations we find

$$-C \wedge \triangleleft C \leq (P = \triangleleft P) \wedge (Q = P) \wedge (\triangleleft P = \triangleleft D)$$

and so, by transitivity,

$$(*) \quad -C \wedge \triangleleft C \leq (Q = \triangleleft D)$$

From the P , Q , and $\triangleleft Q$ equations we find

$$-C \wedge -\triangleleft C \leq (P = \triangleleft P) \wedge (Q = P) \wedge (\triangleleft Q = \triangleleft P)$$

and so, by transitivity,

$$-C \wedge -\triangleleft C \leq (Q = \triangleleft Q)$$

Also, from the Q equation alone,

$$C \leq (Q = \triangleleft Q)$$

Putting these last two lines together we find

$$(-C \wedge -\triangleleft C) \vee C \leq (Q = \triangleleft Q)$$

The left operand can be rewritten

$$(**) \quad -(-C \wedge \triangleleft C) \leq (Q = \triangleleft Q)$$

Putting (*) and (**) together we have

$$\mathbf{if} \ -C \wedge \triangleleft C \ \mathbf{then} \ Q = \triangleleft D \ \mathbf{else} \ Q = \triangleleft Q$$

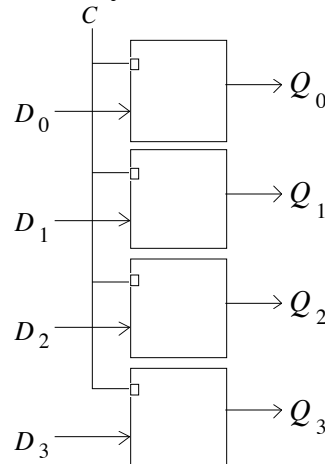
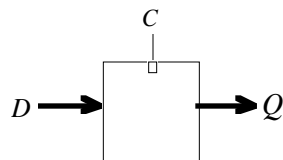
which can be rewritten

$$Q = \mathbf{if} \ -C \wedge \triangleleft C \ \mathbf{then} \ \triangleleft D \ \mathbf{else} \ \triangleleft Q$$

So the master-slave combination is logically equivalent, but more complicated.

Memory

A flip-flop is one bit of memory. We can aggregate flip-flops into a larger amount of memory called a “word”, suitable for storing an **int** or **real** value. We use the same diagram as for a flip-flop but with a thick D input and Q output to indicate many data lines.



All bits in the word are written at the same time, and read at the same time. Algebraically we describe the word by

$$Q_i = \mathbf{if} \ C \ \mathbf{then} \ D_i \ \mathbf{else} \ \triangleleft Q_i$$

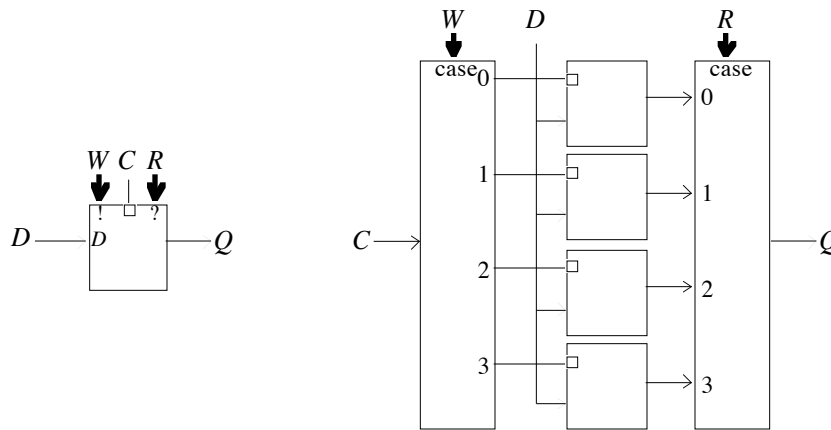
More conveniently, we write

$$Q = \mathbf{if} \ C \ \mathbf{then} \ D \ \mathbf{else} \ \triangleleft Q$$

as before, even when Q and D are several bits each.

The clock input to a word could be rising-edge-triggered, indicated by a small triangle, or falling-edge-triggered, indicated by a small circle and small triangle, the same as for a single flip-flop.

A RAM (random access memory) is an independent way to aggregate flip-flops into a larger amount of memory. One bit is written at a time, and one bit is read at a time. A bit to be written is selected by the writing address W , and a bit to be read is selected by the reading address R .



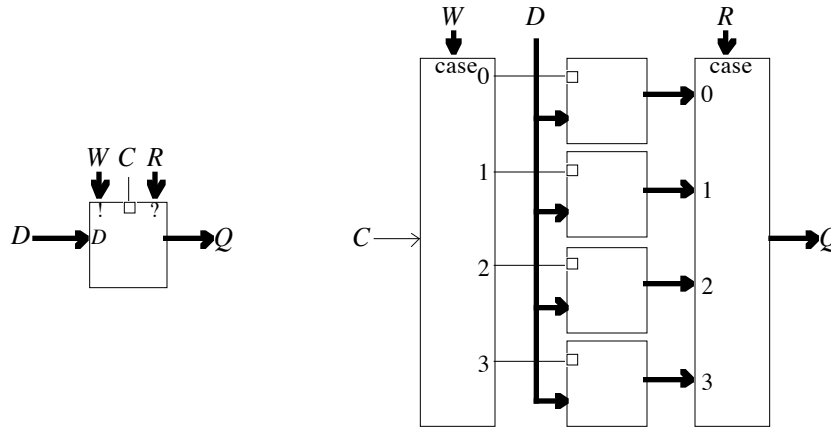
Algebraically,

$$Q = M[R]$$

$$M[i] = \text{if } C \wedge (i=W) \text{ then } D \text{ else } \triangleleft M[i]$$

where $M[i]$ is bit i in the RAM.

The two ways of aggregating bits can be combined to provide a RAM of words. Here are the diagrams.



The clock input to a RAM could be rising-edge-triggered, indicated by a small triangle, or falling-edge-triggered, indicated by a small circle and small triangle, the same as for a single flip-flop.

Merge

A merge turns two sequences of pulses into a single sequence of pulses. (A pulse is a momentary \top). In a sense, any circuit with two inputs and one output is a kind of merge. A \vee -gate allows pulses on either input to pass through; a \wedge -gate allows only simultaneous pulses to pass through.

The 1-2-merge has inputs a and b and output q . It outputs a pulse when pulses arrive on a and b in that order, or simultaneously, but not in the other order. To design a 1-2-merge, we introduce an internal wire A with the meaning “ a is \top or has been \top ”.

$$A = a \vee \alpha \triangleleft A$$

$$q = A \wedge b$$

$$\alpha \leq (\text{pulse time})$$

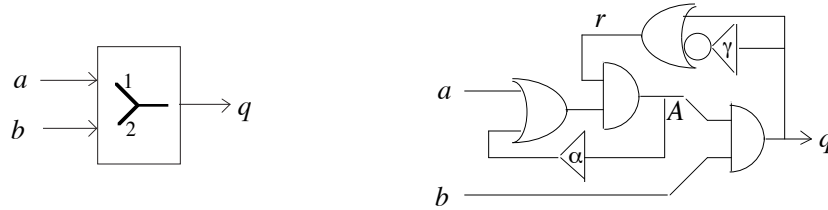
(Recall that $\triangleleft A$ is initially \perp .) Unfortunately this is a one-time-only circuit; if ever there is a pulse on a , it will allow all subsequent pulses on b to pass. To obtain a circuit that resets itself on the falling edge of q ready to be used repeatedly, we introduce one more internal wire r that is \top except at the falling edge of q . The circuit becomes

$$r = q \vee \neg \gamma \triangleleft q$$

$$A = r \wedge (a \vee \alpha \triangleleft A)$$

$$q = A \wedge b$$

$$(\alpha \leq (\text{pulse time})) \wedge (\alpha \leq \gamma)$$



If a pulse on a follows a pulse on b , there must be a delay of at least γ after the end of b before the start of a to avoid truncating the output pulse.

A merge that outputs a pulse when the second of the two input pulses arrives, regardless of their order, and resets itself for reuse, is as follows. The inputs are a and b and the output is q . Internal wire A means “ a is \top or has been \top ”; internal wire B means “ b is \top or has been \top ”; internal wire r is \top except at the falling edge of q . The circuit is

$$\begin{aligned}
 r &= q \vee \neg\gamma \triangleleft q \\
 A &= r \wedge (a \vee \alpha \triangleleft A) \\
 B &= r \wedge (b \vee \beta \triangleleft B) \\
 q &= A \wedge B \wedge (a \vee b) \\
 &(\alpha \leq (\text{pulse time})) \wedge (\alpha \leq \gamma) \wedge (\beta \leq (\text{pulse time})) \wedge (\beta \leq \gamma)
 \end{aligned}$$

