Instant Counters - A Primer for the Ignorant or Forgetful

James R. Drummond

October 1997

Introduction

This is a very fast "run-through" of the relevant properties of digital counters. I am making no attempt to be rigorous or to go through all the details - see a book on Digital Logic for those. I am trying to scope out the territory and draw attention to some details which sometimes aren't highlighted.

The Flip-Flop

The heart of all counters is a one-bit counter called a "flip-flop" or "bistable". This logic circuit can take one of two states and "remember" which one it is in.

The flip-flop will change state to the state of an input line when it is triggered. There are therefore two inputs - the required new state and the trigger.

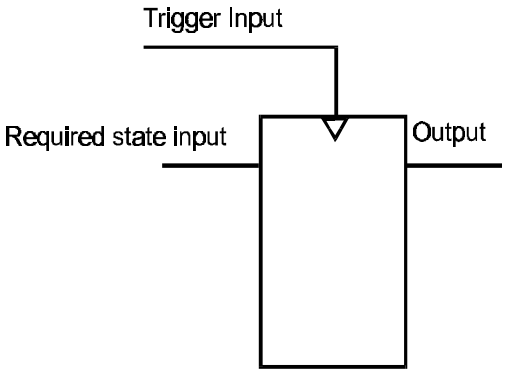

Figure 1 Simple Flip-Flop

A simplified form of the flip-flop may be constructed with the "other" state of the flip-flop as the input, ie if the output is high(low) then the input will be low(high). This only requires a trigger input for operation and will change state on every trigger This is called a "toggle" flip-flop.

The change of state is often initiated by a change of state on an input. Thus a flip-flop may change state when the trigger input moves from the high to the low state. There are other possibilities, but we won't go into them here.

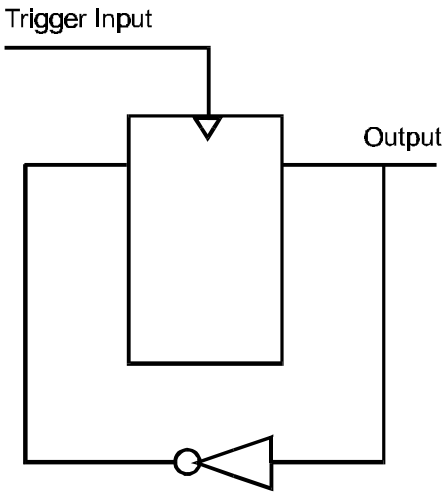

Figure 2 A Toggle Flip-Flop

If we take a simple toggle flip-flop and apply a regular train of high to low transitions to the trigger input then the output will change state on every transition of the input and exactly half those transitions will be high to low transitions. If the input is regular, then so is the output with a rate of exactly one half of the input. If the input frequency is two low to high transitions per second - or 2Hz, then the output will have one high to low per second or 1Hz. Thus the toggle flip-flop is often referred to as a

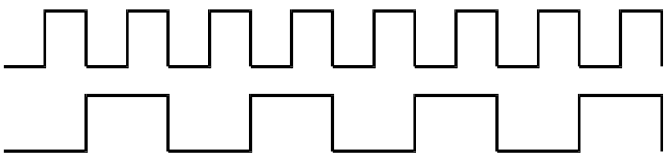

Figure 3 a divide-by-2 sequence

“divide by 2" circuit. It is this property which can be used to construct counters because the output can be considered as “counting in 2s” which is the second digit of a binary number. If we further interpret the input and output as the Least-significant-bit (LSB) and Most-Significant-Bit (MSB) of a two-bit counter we can also interpret the pair of bits as counting 00, 01, 10, 11, 00, etc. Which is simple binary count sequence.

Ripple Counters

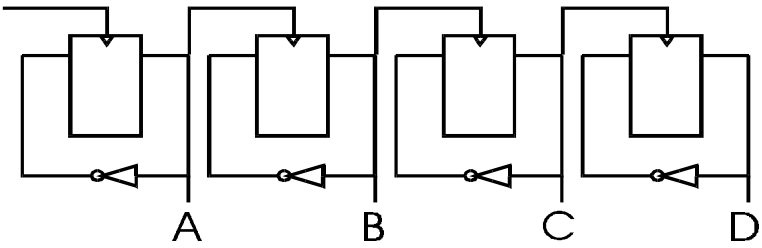

Figure 4 A binary ripple counter

Since the output of the toggle flip-flop looks the same as the input - we can use that as the input to another toggle flip flop and so on down the line. This will construct a “divide-by-2^n” circuit or an n-bit counter depending upon how we interpret the outputs. Notice that my outputs now don’t include the input line - this is just a matter of convenience and you could just as easily include the input.

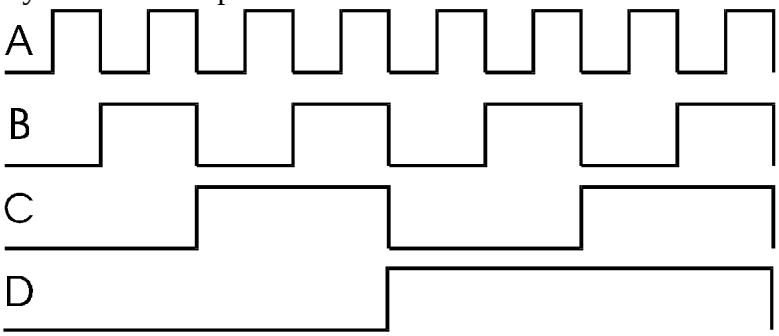

Figure 5 Outputs for the above counter

Although I have drawn the input as a regular succession of edges, there is obviously no reason why an irregular rate of edges should not have the same effect, so this is a true counter.

The main problem with ripple counters is that life is not as simple as the diagrams. It actually takes time for a ripple counter to make a transition and so although the set of transitions in the middle of the diagram above all appear to be one above the other, ie co-incident in time, they aren’t. The sequence is from the top to the bottom, from the least-significant to the most significant. Thus the succession is 0111, 0110, 0100, 0000, 1000, whereas we thought it went from 0111 to 1000 directly! This may seem like a “picky” point, but it is quite significant. If you choose to read the state of a ripple counter while it is counting, you might not get anything like the right answer! For example if the input clock is 1Mhz - with an identical rate of high to low transitions, and a flip-flop takes 10nS to make a transition, there is a 0.2% chance of an error. Small, but not zero. Notice that there are only problems when two outputs are required to change “at once”, there is no problem when only the LSB changes.

However ripple counters are simple, cheap, and reliable if you don’t read them while they are counting.

Synchronous Counters

I don't have much space to go into synchronous counters here - but I will attempt to cover the high points. With a ripple counter, the input is only applies to the first flip-flop. In a synchronous counter the input is applied to all flip-flops. The changes of state are handled by making the individual flip-flop inputs the correct ones. This means we don't use toggle flip-flops (except for the first stage) and we need more logic gating. The essential plan is that we know what state we are in (by looking at the outputs), we know what state we need to get to (count plus one), we know what inputs would be required to get to that state on the next transition and we arrange the logic to provide those inputs.

The logic can be simple or complex. For a simple binary counter the logic is to AND all less significant outputs and EXCLUSIVE OR the result with the current output. (If you aren't familiar with logic terminology it is sufficient at this stage to recognise that there is a formula)

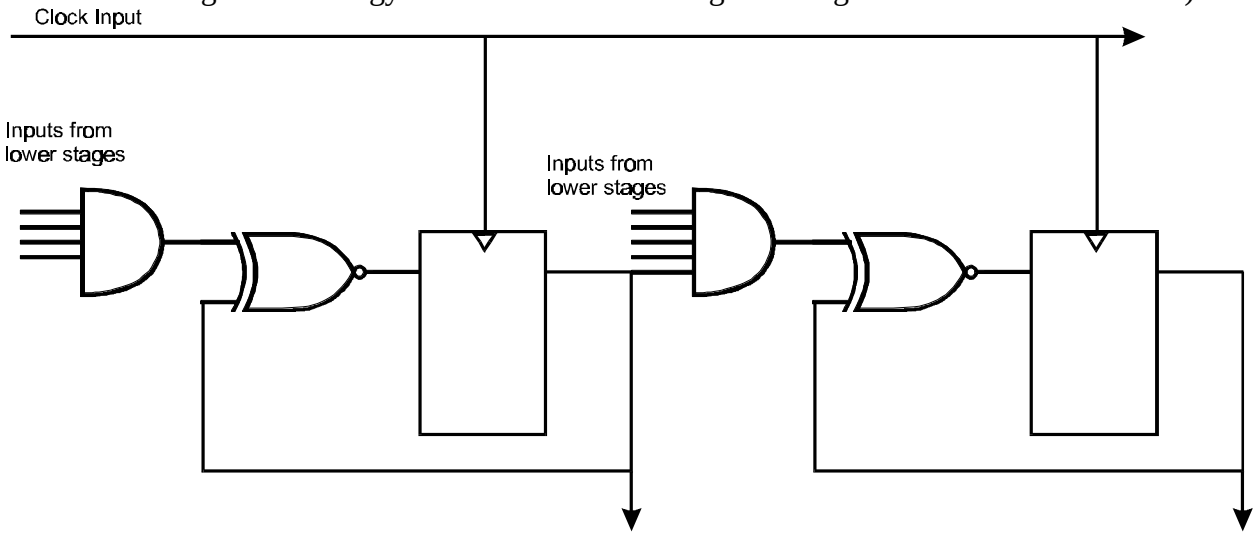

Figure 6 Two stages of a synchronous binary counter

The advantage of a synchronous counter is that it can be read while it is counting since all the outputs change at once. The disadvantage is that it is more complex and ultimately slower because of the additional logic burden.

The other advantage of a synchronous counter, which I won't go into here, is that it doesn't have to be a binary counter. The count sequence is arbitrary - a simple "up" counter is just one of the possibilities.

Phrased in more elegant terms, a synchronous counter consists of a set of flip-flops which executes a certain set of states in which the next state is predicted by the current state. This means that an arbitrary counter can be drawn in the following terms where the "logic block" is what sets up the inputs for the next state, given the current state.

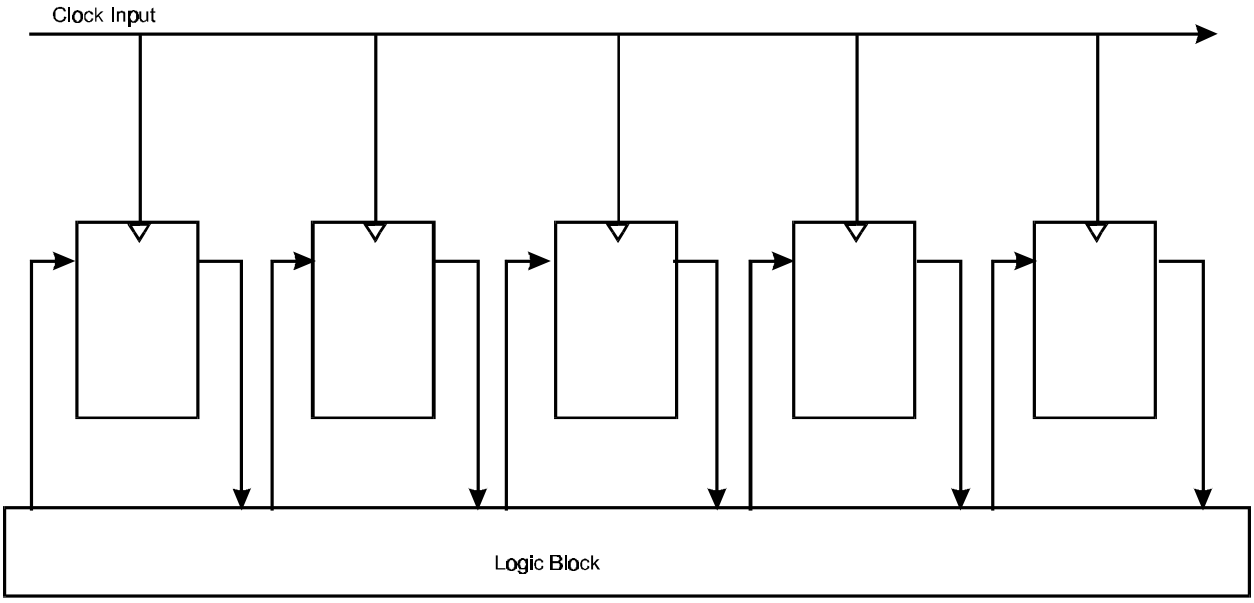

Figure 7 A General Synchronous Counter

The advantage os a synchronous counter is that it does not have any invalid or “ripple states” (although you might like to think about whether the statement that “all the stages change together is an accurate one”). The disadvantage is that the count rate will be determined by the time that it takes the logic block to set up the next set of inputs. This time is quite short, but it isn’t negligible.