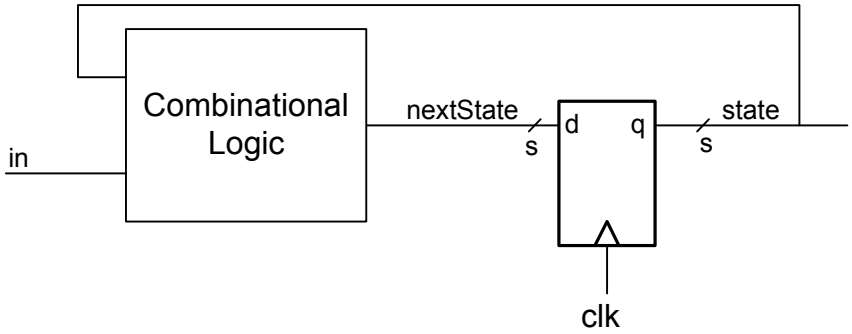
CS99S
Laboratory 5 Preparation

October 22, 2001

Objectives:

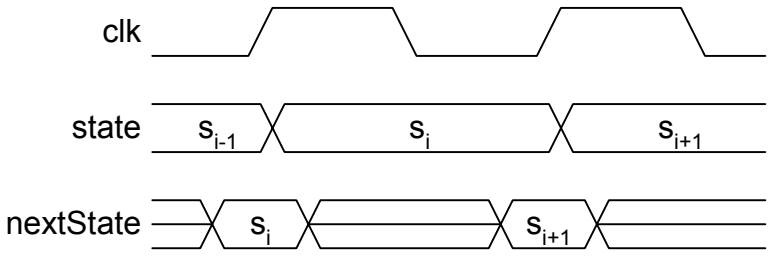
1. Understand the principles of synchronous sequential logic
2. Build a quadrature counter from flip-flops

Synchronous Sequential Circuits



Using D flip-flops, we can build sequential logic circuits that have the form shown in the figure above. The combinational logic computes the *next state* of the circuit, an s-bit vector as denoted by the slash, as a function of the present *state* and the input. We can write this as nextState = f(in, state) for some function f.

The s D flip-flops (one for each bit of the state vector) act to *synchronize* the transition from one state to the next so all of the bits of state change at the same time, on the rising edge of the clock. The timing diagram below illustrates this synchronization. Each rising edge of the clock samples the next state and updates the state with its value. Thus, all state bits change just once and at the same time, just after the rising edge of the clock, regardless of how fast or slow the individual paths through the combinational logic are[1] or whether some of the next state bits may *glitch* to incorrect values before reaching their final values. This synchronization allows us to think of the operation of this circuit in terms of a sequence of states $s_1, s_2, \ldots s_n$ where $s_i = f(in_{i-1}, s_{i-1})$ and not bother with the exact details of circuit timing.



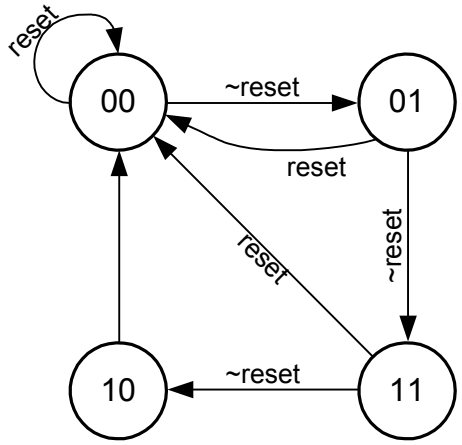
[1] As long as the paths are fast enough so that all next state bits are computed before the rising edge of the clock.

A Quadrature Counter

The next state function, f, of a synchronous sequential circuit can be expressed by a *state table* that gives the next state for each combination of input and present state. Consider, for example, a quadrature counter. This circuit has a single input, reset, and two state bits. Asserting reset high clears the state bits on the next rising edge of clock. When reset is low, the state bits cycle through the four states 00, 01, 11, 10, advancing one state on each rising edge of the clock. The state table for the quadrature counter is shown below:

reset	state	nextState
0	00	01
0	01	11
0	11	10
0	10	00
1	**	00

Sometimes it is more descriptive to write our state table in a graphical form called a *state diagram*. In a state diagram, each state is drawn in a circle and the transition from one state to the next state is indicated by an edge labeled with the input condition that causes that transition to occur. A state diagram for the quadrature counter is shown below.



In each state, the edges out of that state show what the next state will be if reset is true (all of these arrows go to state 00) or if reset is false (these arrows proceed around the square). The reset edge out of state 00 leads back to state 00. This implies that as long as reset is true we stay in state 00 – the next state is the same as the present state. The arrow out of state 10 has no label. This means that this edge is always taken regardless of the value of the input.

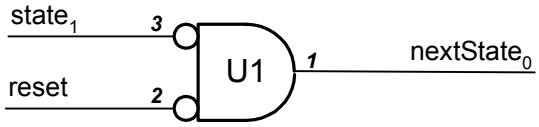
From the state table (or state diagram), we can write down a truth table for each of the two next state bits. From this truth table, in turn we can derive the logic function for that state bit. For example, the truth table for the least significant state bit, $nextState_0$ is:

reset	$state_1$	$state_1$	$nextState_0$
0	0	0	1
0	0	1	1
0	1	1	0
0	1	0	0
1	*	*	0

From this table, we can write down the logic function for $nextState_0$ as:

$nextState_0 = \sim reset \wedge \sim state_1$

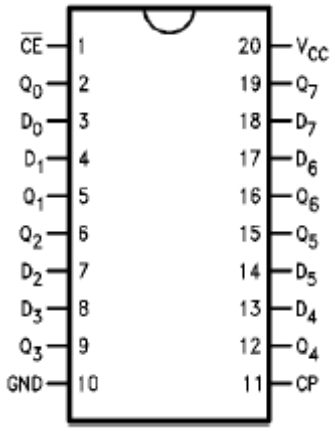
From this equation we can draw a logic diagram that computes this next state bit using a 74AC01 NOR gate (U1). (Remember a NOR is equivalent to an AND with inverted inputs.)



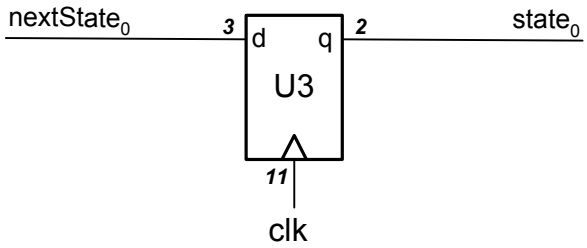
Prep Question 1: Write down the truth table, logic function, and logic diagram to compute the most significant bit of the next state function, $nextState_1$. Make sure your schematic is labeled and don't use the same gate that was used to compute $nextState_0$ above (you can use unused gates on chip U1 though).

The 74AC377 Octal D Flip-Flop

Now that you have the logic circuits for both bits of the next state function, all you need to complete your quadrature counter is some flip-flops. Rather than build these from NAND gates, we will use a 74AC377 octal flip-flop part (U3). The pinout for this part is shown below. To use the part you must connect up the power supplies (pin 10 to GND and pin 20 to V_{DD}), tie the clock enable low (pin 1 to GND), and supply the clock to the CP pin (pin 11). You can use either a debounced pushbutton or your relaxation oscillator (both of which you should have built in lab 1) for your clock.



Using one of the flip flops on this part, the schematic for generating bit 0 of the state vector from bit 0 of the next state vector is:



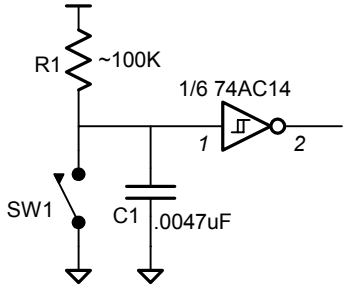
Prep Question 2: Draw a schematic diagram for the entire quadrature counter circuit. This should include the combinational logic for both next state bits and both flip flops. You should have this all in a single diagram so you can wire from it.

The Lab

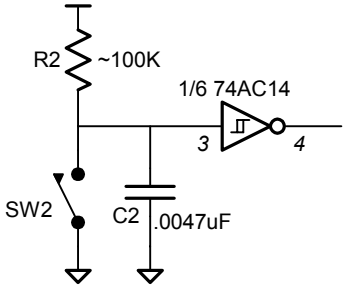
For the lab you should wire up your quadrature counter from the schematic you drew for prep question 2. Once your counter is wired, hook the clock to your debounced pushbutton and step your counter through its states to verify its operation.

Appendix – Debounced Pushbutton and Relaxation Oscillator

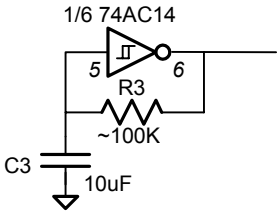
Since many of you did not finish this part of Lab 1, the schematic for these circuits are repeated here. You only need one of the debounced pushbuttons for the lab.



Debounced pushbutton 1



Debounced pushbutton 2



Relaxation Oscillator

The debounced pushbutton takes one of your pushbutton switches and *debounces* it. When a mechanical switch changes state, it often *bounces* alternately making and breaking contact until it comes to a stable on or off state. If we were to directly use a pushbutton to clock our quadrature counter, the counter might advance more than one state on each press of the button as the clock signal goes high and low during the bounce. To debounce our pushbutton we put a capacitor on the output of the switch so the resistive pullup occurs slowly (with a time constant set by RC to be about 470 microseconds. We then put this filtered signal into a 74AC14 Schmidt trigger inverter which has hysteresis. The combination of the low-pass filter and the hysteresis eliminates the switch bounce giving us a single transition each time the pushbutton is pressed.

Of your finger gets tired of pressing the button every time you need a clock pulse you can use the relaxation oscillator instead. This oscillator uses negative feedback around a Schmidt trigger gate to generate a square wave. If the gate input is low, the output goes high and starts charging the capacitor up. As soon as the capacitor charges above the top end of the hysteresis, the output goes low and starts discharging the capacitor. When the capacitor discharges below the bottom end of the hysteresis range the output will go high again. The process repeats with the time constant for each interval being set by RC to be about 1 second.